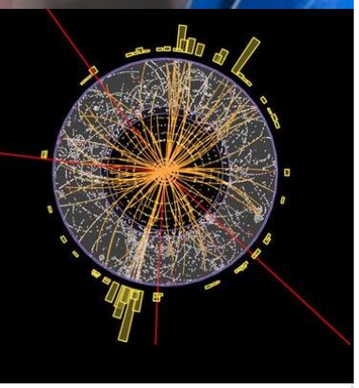
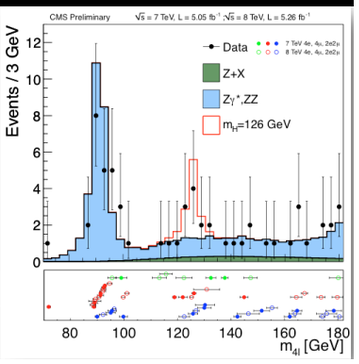
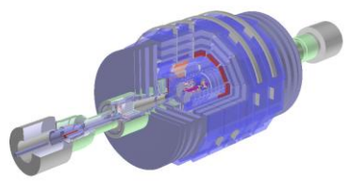




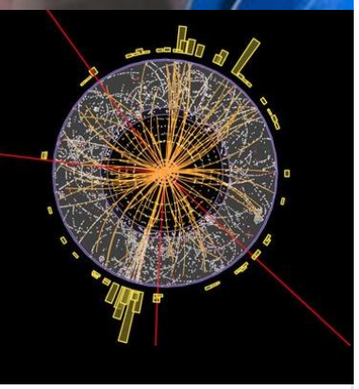


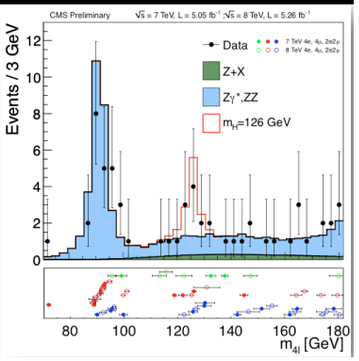
ROOT 6 and beyond: TObject, C++14 and many cores.

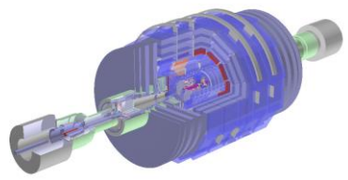
Philippe Canal
Fermilab

On behalf of ROOT Team.

Overview

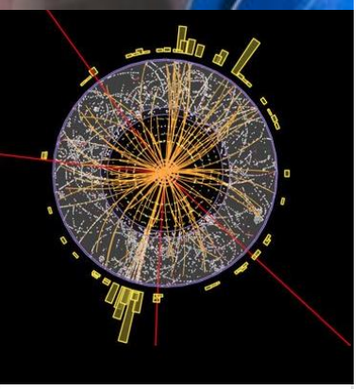


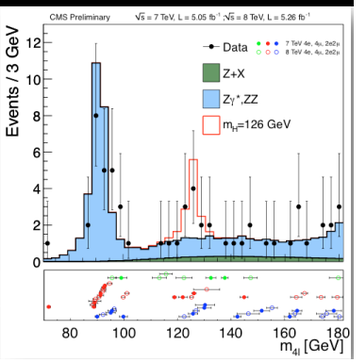


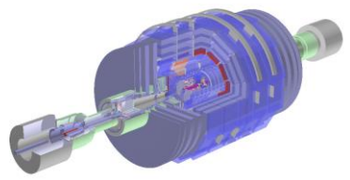


- Current releases
  - Core, Cling
  - Threading, I/O
  - Graphics, Network
- v6+ and beyond
  - Why & what

ROOT @ CHEP







- Poster session A

Base ROOT reference guide on Doxygen

Presented by **Olivier COUET**

- Poster session B

Deep Integration: Python in the Cling World

Presented by **Wim LAVRIJSEN**

JSROOT version 3 – JavaScript library for ROOT

Presented by **Dr. Sergey LINEV**

THttpServer class in ROOT

Presented by **Dr. Sergey LINEV**

- Track 4 Session

16 Apr 2015 at **09:30**

ROOT6: a quest for performance

Presented by **Danilo PIPARO**

16 Apr 2015 at **09:45**

ROOT 6 and beyond: TObject, C++14 and many cores.

Presented by **Philippe CANAL**

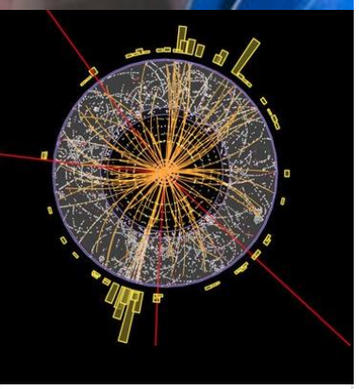
- Track 2 Session

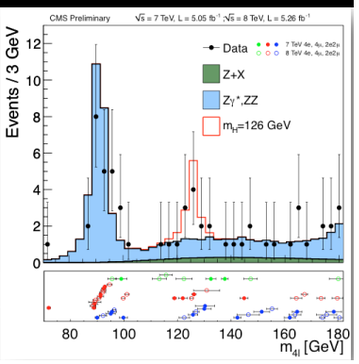
16 Apr 2015 at **11:00**

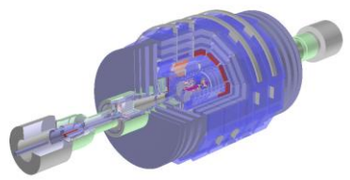
Using R in ROOT with the ROOT-R package

Presented by **Lorenzo MONETA**

Core



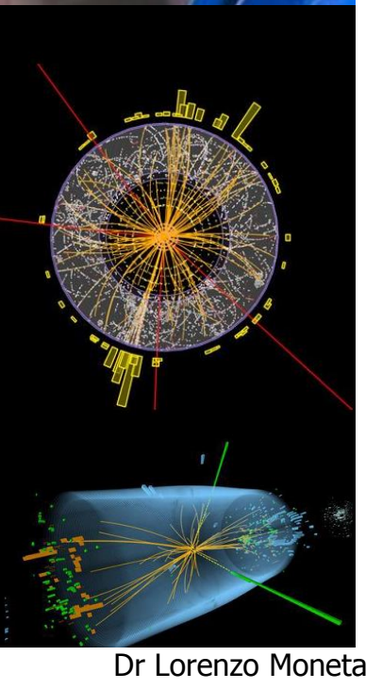




- ***Cling***
  - Replaces ***CINT***: a radical change at the core of ***ROOT***
  - Based on ***LLVM*** and ***CLANG*** libraries.
  - Full support for ***C++11/14*** with carefully selected extensions for ease of use
  - Script's syntax is much stricter
  - ***I/O*** fully backward and forward compatible
  - Will allow support for more architectures (***ARM64***, ***PowerPC64***)
- ***C++11*** or higher now required (and used within ***ROOT***)
- ***CMake*** becoming main build system.

v6.04

New TFormula

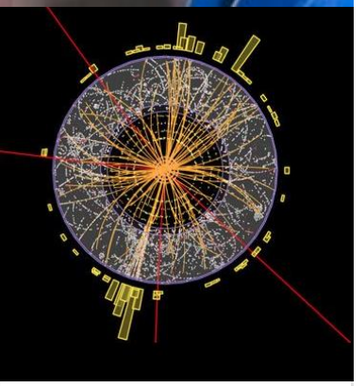


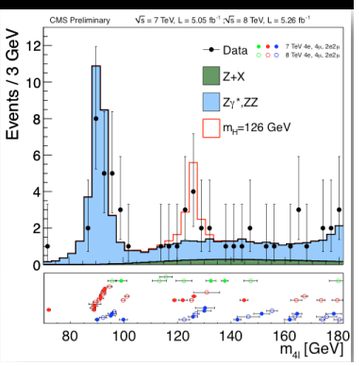
- ***TFormula*** class re-written to use Cling for compiling expressions
  - correctness of results
  - use compiled code for evaluating expression
    - same performance as C++ compiled functions
  - easy extendable with extra functionality
    - e.g. adding new pre-defined function
    - use of a vectorised interface
- Old ***TFormula*** still available as ***ROOT::v5::TFormula***
- **3x** faster for typical expressions

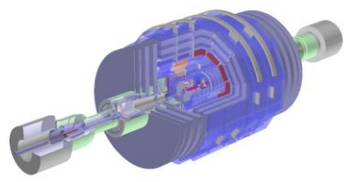
```
exp(-0.5*(x-[1])/[2])^2)
```

Dr Lorenzo Moneta
CERN PH-SFT
CH-1211 Geneva 23
sftweb.cern.ch
root.cern.ch

Threading







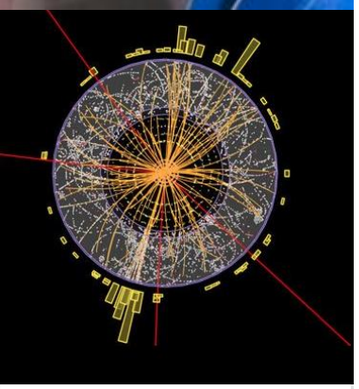
- Focus has been ***so far*** on:
  - 1 file/1 tree per thread/stream of operation
  - ***Core, Meta, I/O, Tree, Hist*** and ***Fit*** libraries

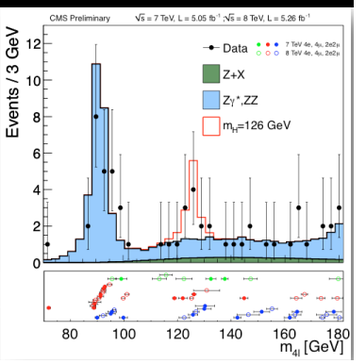
Thanks to Chris Jones

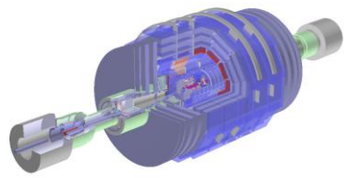
- Main beta tester and provider of fixes: ***CMS***
  - Error rate in production jobs was negligible

- ***Helgrind*** and static analysis, and routine testing essential
  - False positive hidden by:
    - *$ROOTSYS/etc/valgrind-root.supp*
    - *$ROOTSYS/etc/helgrind-root.supp*

I/O







- Update to ***I/O*** meta data to support cling
- Extensions to support more cases of schema evolution

- Introduced ***TTreeReader***:

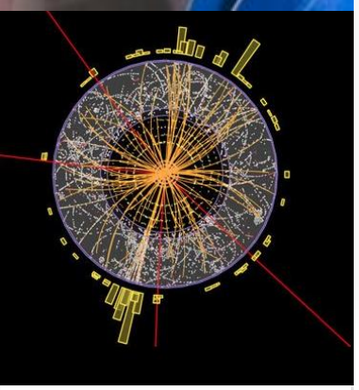
No SetBranchAddress or GetEntry

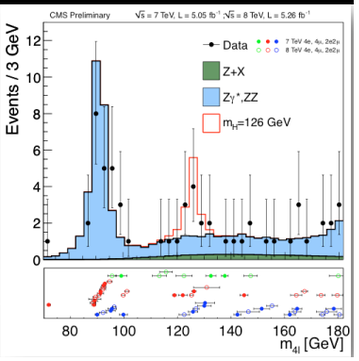
```
TFile *f = TFile::Open("tr.root");
TTreeReader tr("T");
TTreeReaderValuePtr<MyPart> p(tr, "p");
while (tr.GetNextEntry()) {
   printf("Momentum: %g\n", p->GetP());
}
```

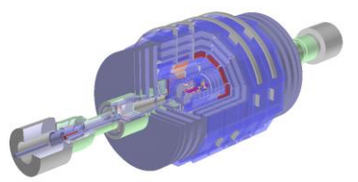
- ***I/O*** for ***C++11*** containers:
  forward_list, unordered_set, more on the way
- ***TTreeCache*** enabled by default (v6.04)
  - Prefill also turned on
    - A single read of the cluster for **all** branches during *TTreeCache* training phase
    - Remove the many small reading during training

Thanks to David Smith and Peter Van Gemmeren

Graphics







TMathText
Thanks to
Yue Shi Lai

$$\prod_{j\geq 0}\left(\sum_{k\geq 0} a_{jk} z^k\right) = \sum_{n\geq 0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\geq 0 \\ k_0+k_1+\cdots=n}} a_{0k_0} a_{1k_1} \cdots \right)$$

$$W^{3\beta}_{\delta_1\rho_1\sigma_2} = U^{3\beta}_{\delta_1\rho_1\sigma_2} + \frac{1}{8\pi^2}\int_{a_1}^{a_2} da_2' \left[\frac{U^{2\beta}_{\delta_1\rho_1} - a_2' U^{1\beta}_{\rho_1\sigma_2}}{U^{0\beta}_{\rho_1\sigma_2}} \right]$$

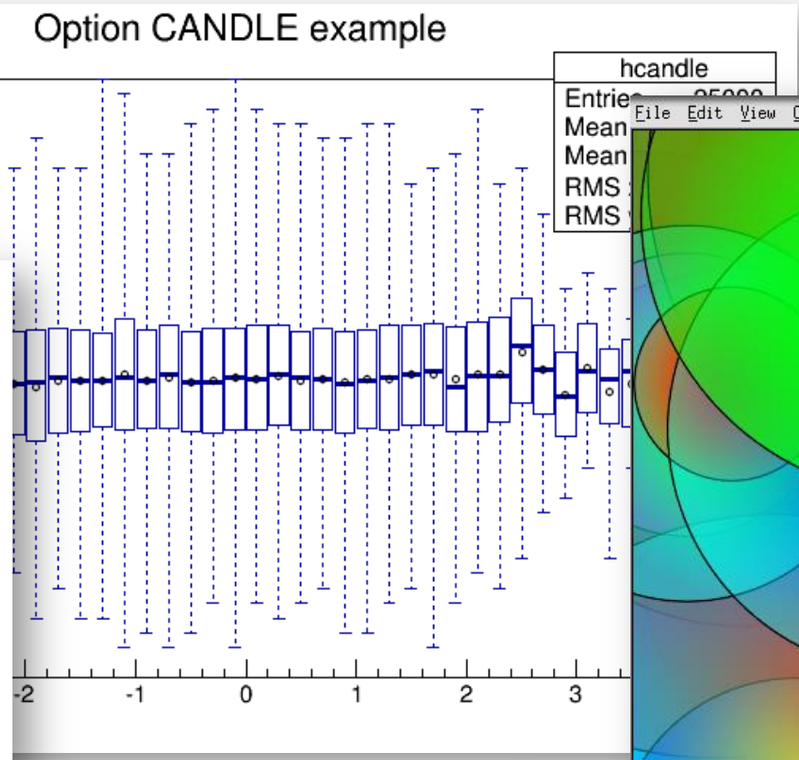
$$d\Gamma = \frac{1}{2m_A}\left(\prod_f \frac{d^3p_f}{(2\pi)^3}\frac{1}{2E_f}\right)|\mathcal{M}(m_A - \{p_f\})|^2 (2\pi)^4 \delta^{(4)}(p_A - \sum p_f)$$

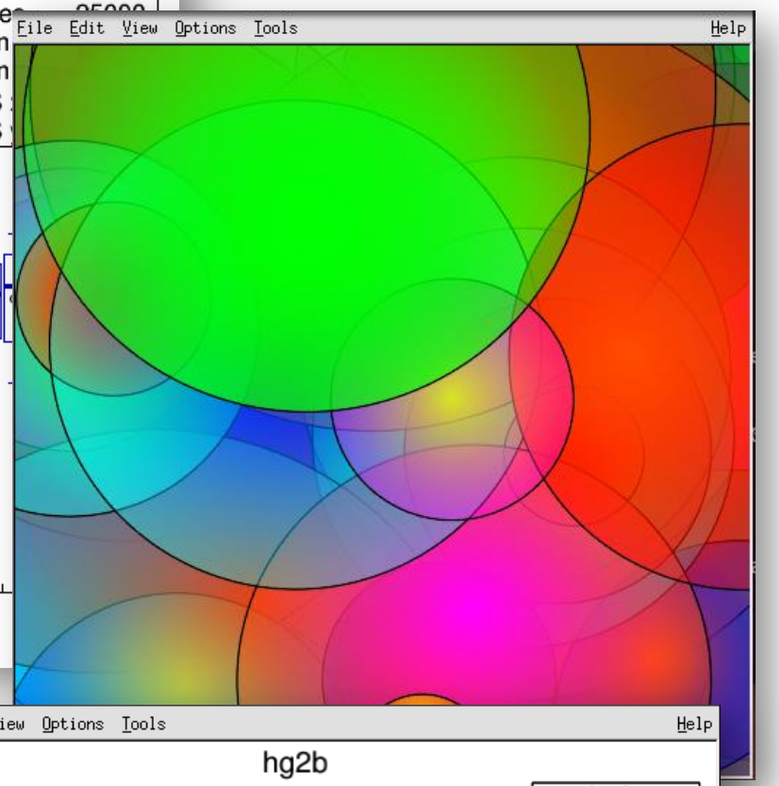
$$4\mathrm{Re}\left\{\frac{2}{1-\Delta a}\chi(s)\left[\hat{g}_v^e \hat{g}_v^f (1+\cos^2\theta) + \hat{g}_a^e \hat{g}_a^f \cos\theta\right]\right\}$$

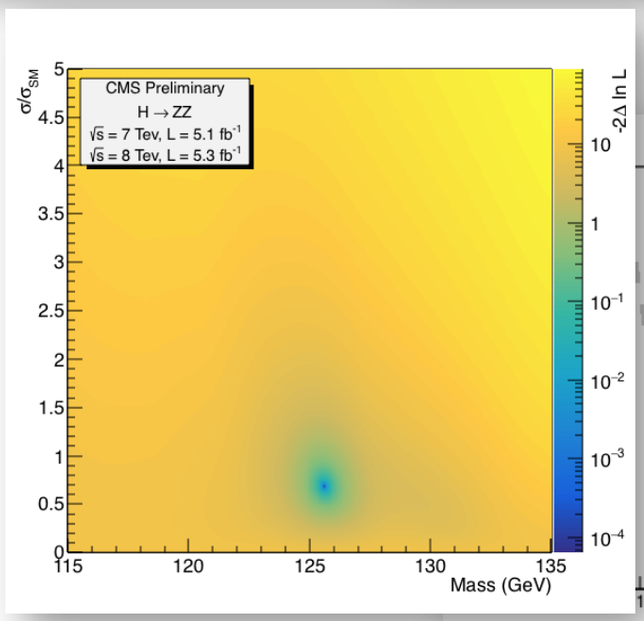
$$p(n) = \frac{1}{\pi\sqrt{2}}\sum_{k=1}^{\infty}\sqrt{k}A_k(n)\frac{d}{dn}\frac{\sinh\left\{\frac{\pi}{k}\sqrt{\frac{2}{3}}\sqrt{n-\frac{1}{24}}\right\}}{\sqrt{n-\frac{1}{24}}}$$

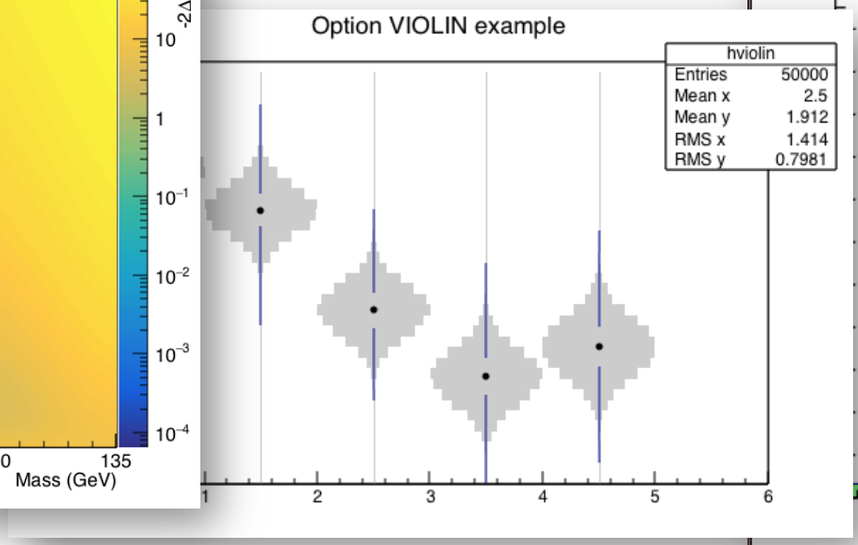
$$\frac{(\ell+1)C_\ell^{TE}}{2\pi} \quad \mathbb{N}\subset\mathbb{R}$$

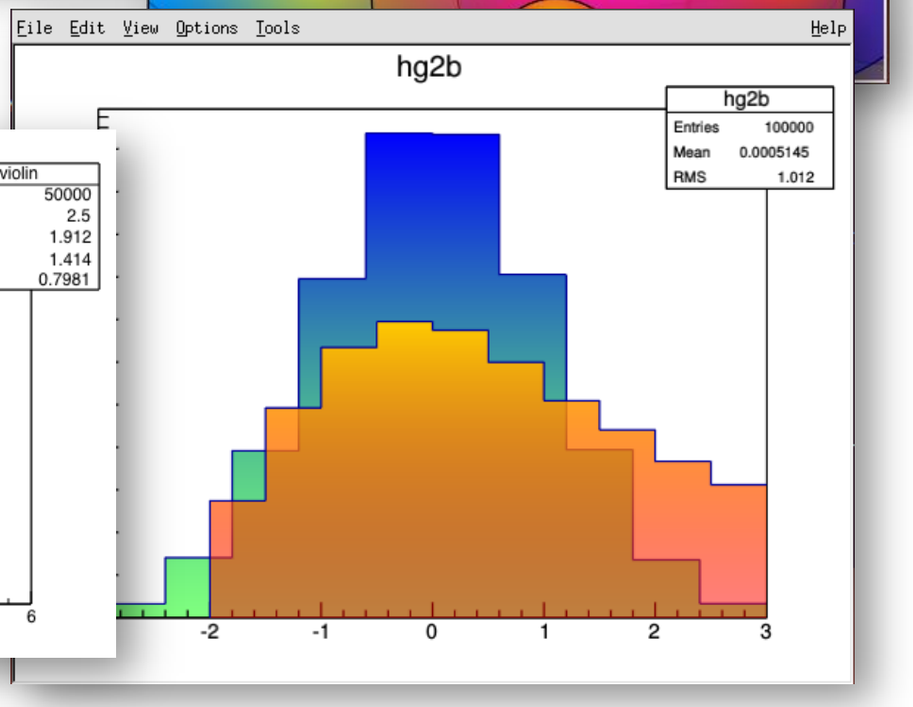
RHIC スピン物理 Нью-Йорк











Say No To Rainbow Palette

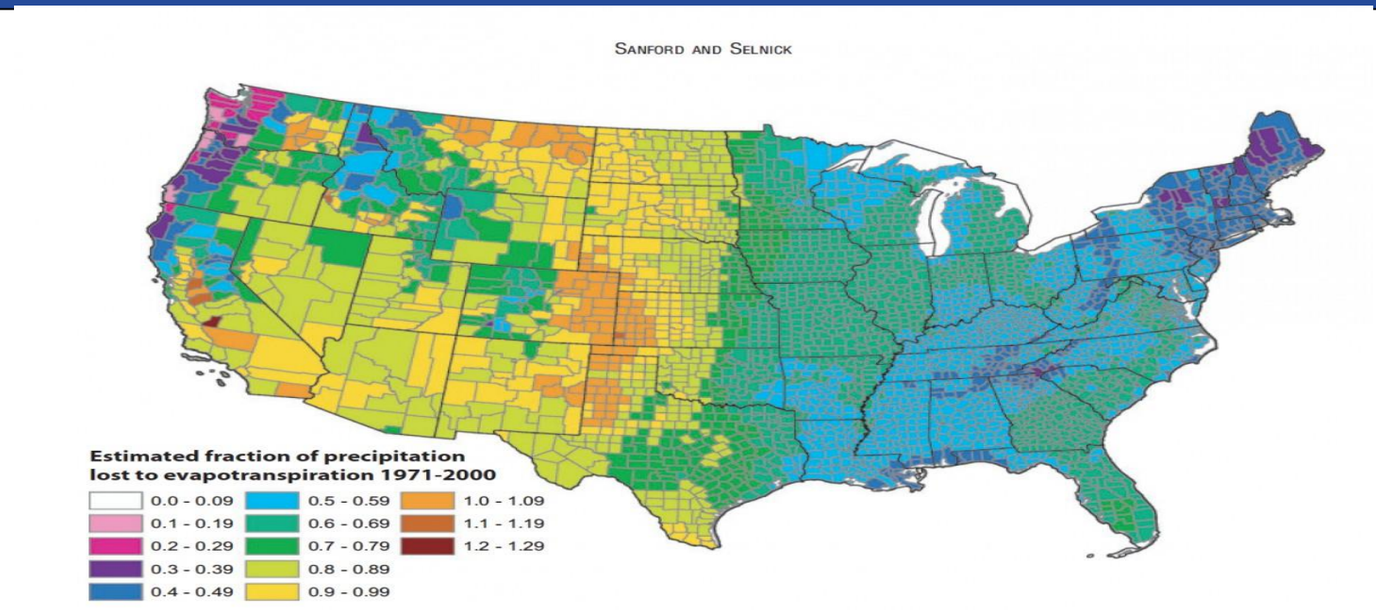
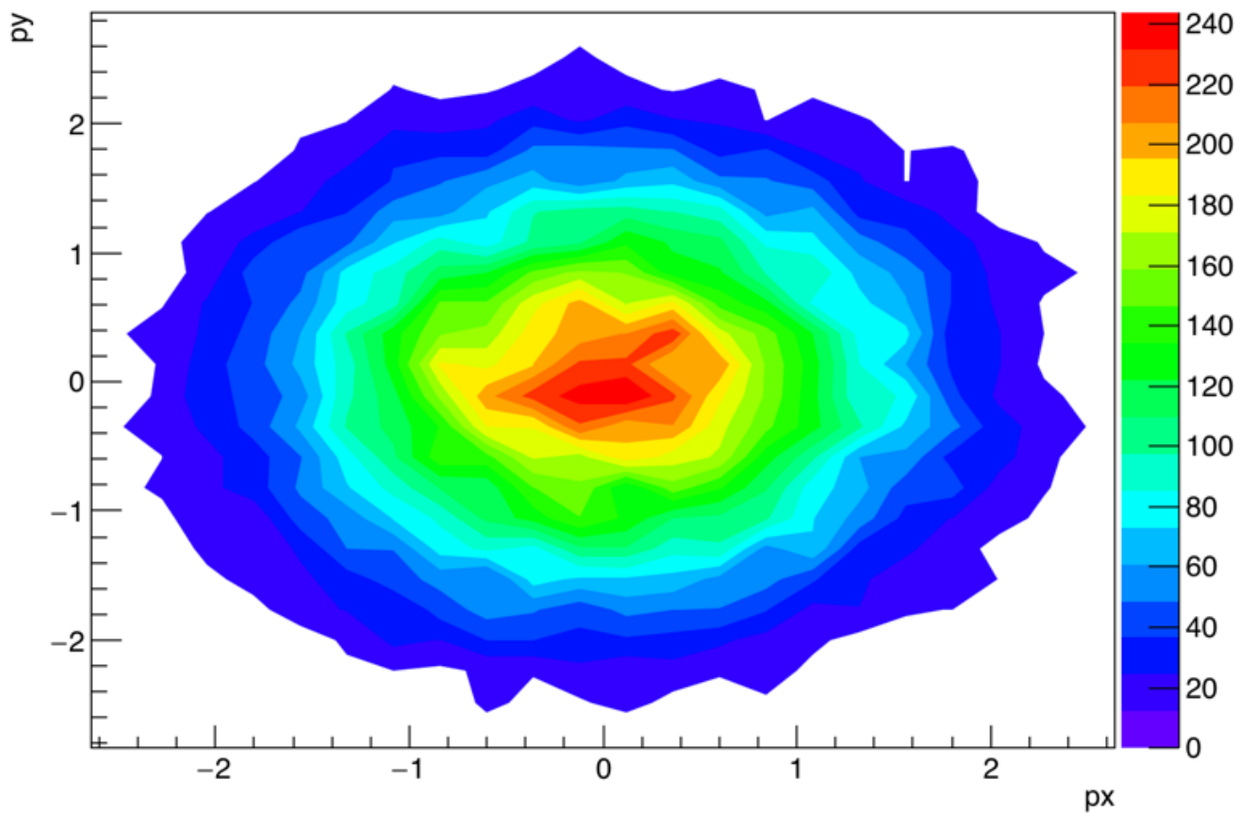


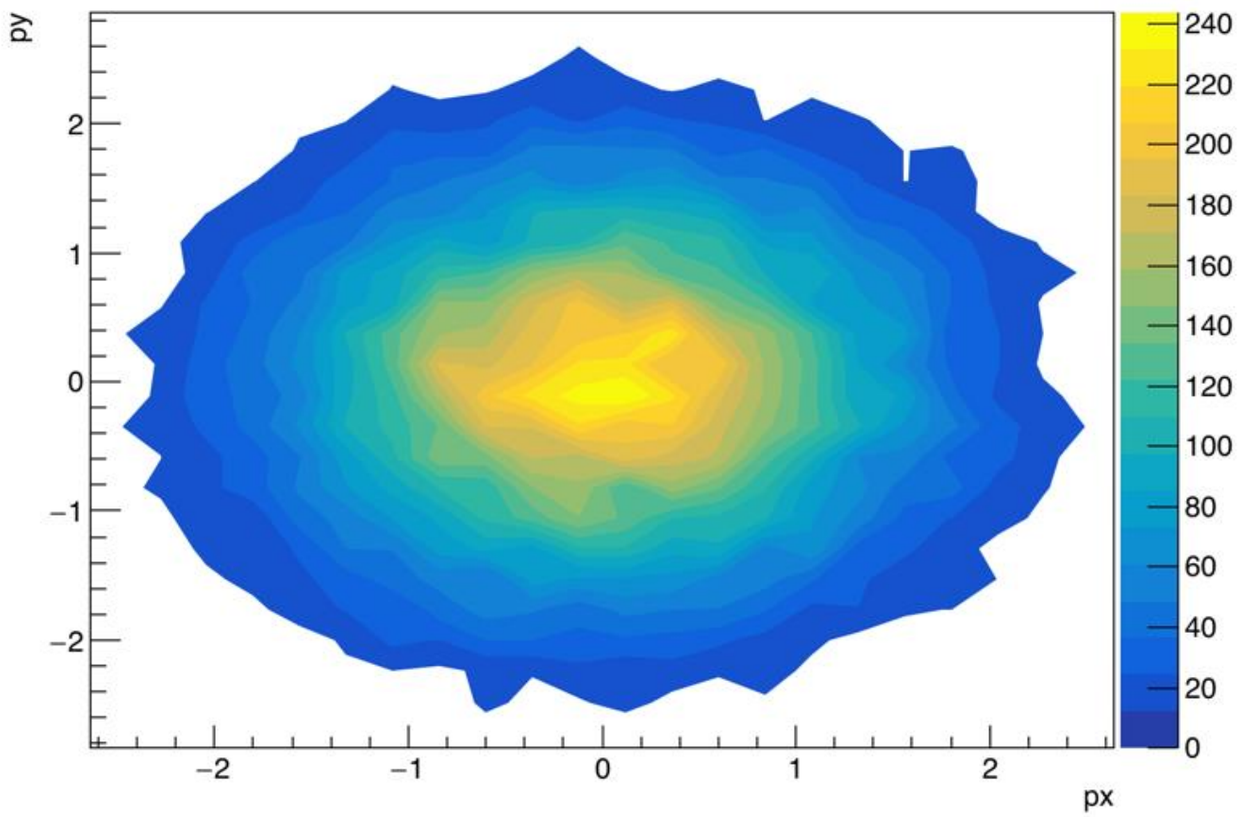
FIGURE 13. Estimated Mean Annual Ratio of Actual Evapotranspiration (ET) to Precipitation (P) for the Conterminous U.S. for the Period 1971-2000. Estimates are based on the regression equation in Table 1 that includes land cover. Calculations of ET/P were made first at the 800-m resolution of the PRISM climate data. The mean values for the counties (shown) were then calculated by averaging the 800-m values

```
// Use new palette with:
gStyle->SetPalette(57);
```

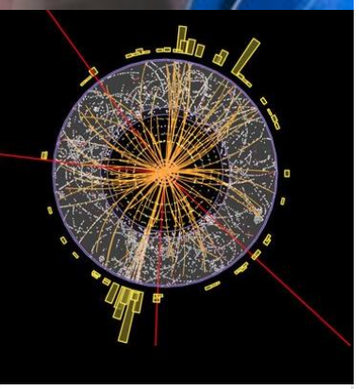
py:px {px*px+py*py < 20}

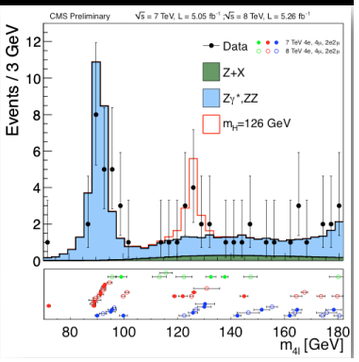


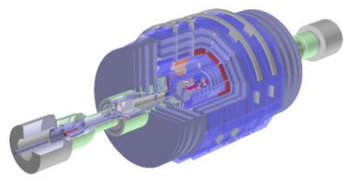
py:px {px*px+py*py < 20}



Remote accesses



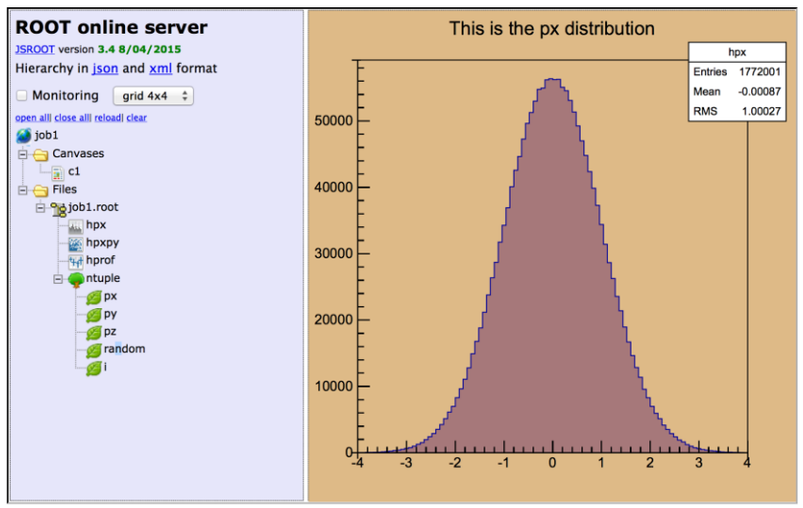




- New plugin based on ***Davix*** is the default for http file access
- ***JSON*** is now a supported ***I/O*** backend
- ***JavaScript*** library ***JSRootIO***
  - Continue support and on-demand extension including drag and drop, context menus, etc.
- ***HttpServer*** introduced for easy web display development:

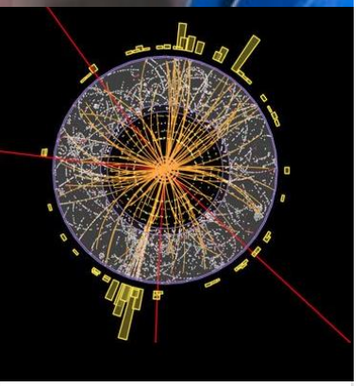
Thanks to Sergei Linev

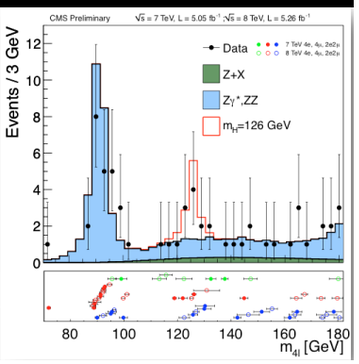
```
THttpServer serv;
serv.Register("abc/fold1", hpx);
// Then browse via http://localhost:8080
```

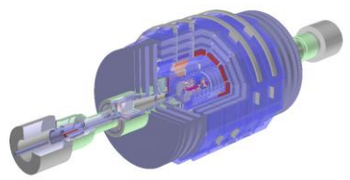


  - Can be used for custom (live) display by using ***javascript*** and ***JSRootIO***

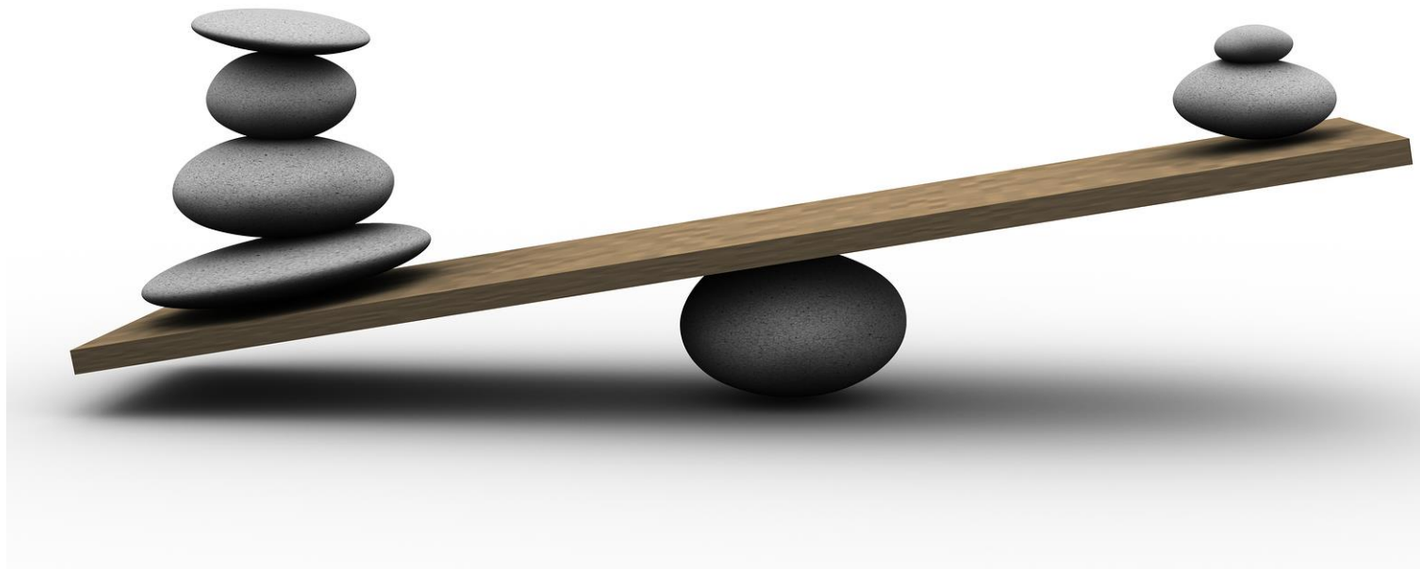
v6+ and beyond





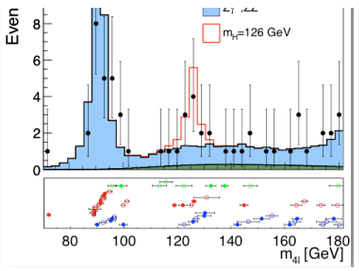


- Leveraging of
  - ***Cling***
  - ***C++11/14/17***
  - And 20 years of experiences providing solutions to a wide range of users

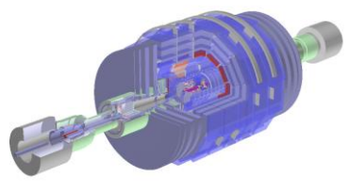


To what end?

- Increase overall performance
- Increase user friendliness

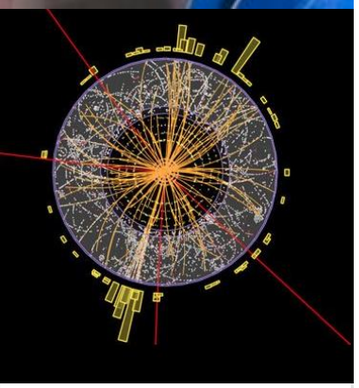


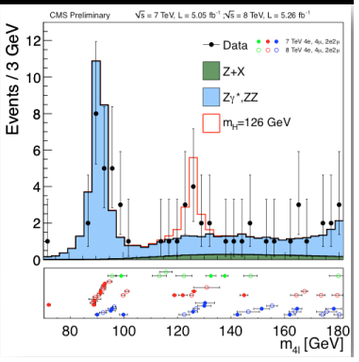
- Clarify and leverage parallelism with ***ROOT***

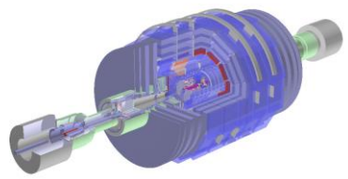


- Standardization

Increase overall performance

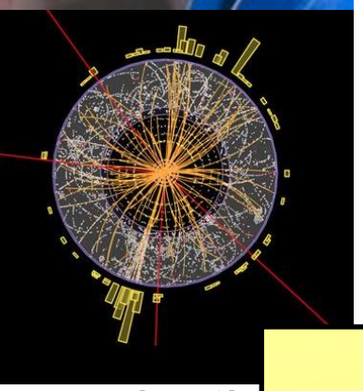


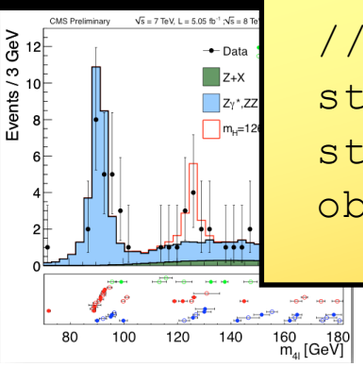


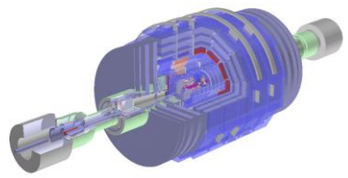


- Vectorization
  - Careful inlining, data structure improvement
  - Fitting, ***RooFit***, ***I/O***, ***Hist***
  - Commonly used Math functions
- JIT compilation in ***TTreeFormula*** and ***I/O***
- Reduce virtual interfaces; move runtime checks to compile time options
- ***I/O*** runtime, disk space, memory improvements
  - switch to little-endian, compress each entry individually to improve random access, reduce cost of repeated [deep] hierarchies
- ***TTreeCache***
  - New ***OptimizedBasket***, prefetching algorithms

Increase user friendliness







- Many interfaces can be improved in ***C++14,17***
  - Ownership, type safe containers, string options
  - Resulting in improved user productivity
    - Dramatically reduce memory errors, wrong results, etc.
  - Code Self-documentation

```
void OwnOrNot(std::unique_ptr<TWhatever> arg);
 OwnOrNot( & myWhatever ); // Compilation error!
```

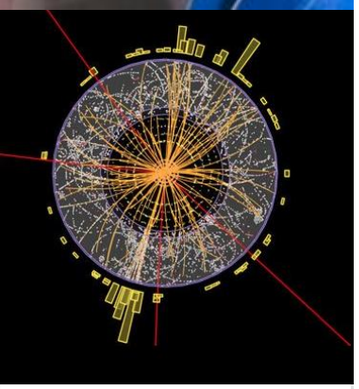
```
// With SetName(const char*)
std::string str; …
std::string sub( str.data()+pos, len );
obj.SetName( sub.c_str() );
```

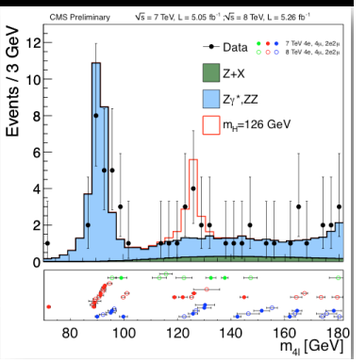
```
// With SetName(std::string_view)
std::string str; …
obj.SetName( {str.data()+pos, len} );
```

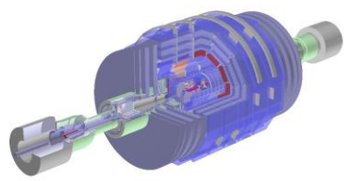
```
// Current
TFile f(name); TH1F h1(…);                    f.Write();

// ROOT v7, no implicit shared ownership.
TFile f(name); auto h1 = f.Create<TH1F>(…); f.Write();
```

Clarify and leverage parallelism

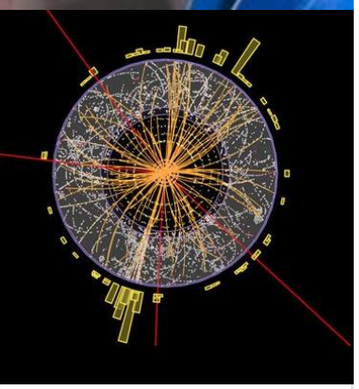


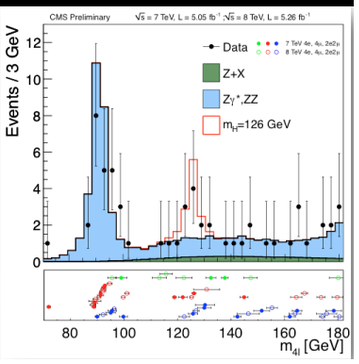


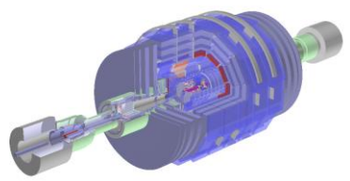


- Clarify concurrent access capability of APIs
  - Designing new interfaces when necessary
- Add scheduling interface to allow coordination with framework's task scheduler
- Use multiple threads/tasks in:
  - ***Histogram***, ***TTree***, ***I/O***, ***Math***
- Make it easy to write analysis using multi and many cores
  - With ***efficient*** merge step
  - Build on the strengths of ***PROOF(-Lite)***
- Make it easier to transition existing code
  - Provide hints of required changes (CMS static analyzer ext.)

Standardization



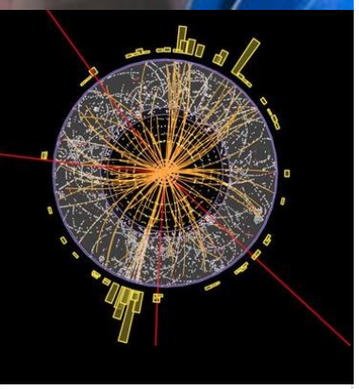


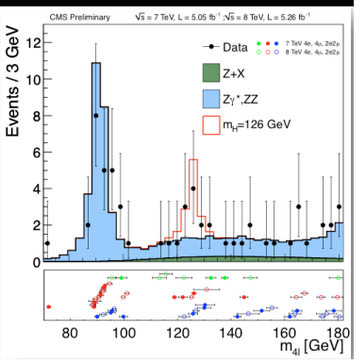


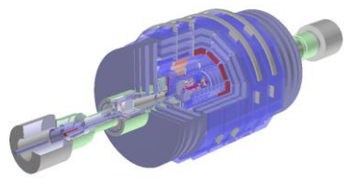
- Extent support for and more extensively use of new ***C++11*** constructs
  - ***std::string***, ***std::string_view***
  - ***std::array***, ***std::shared_ptr***, ***std::unique_ptr***
  - new ***STL*** collections.

- Continue to explore, standardize and offer ***HEP*** common solutions

Interfaces Revolution – v7







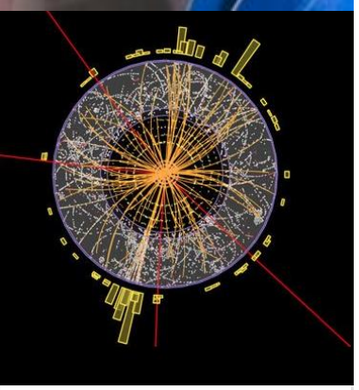
- Large existing code base relied upon in production across sciences and continents
  - Must be backward compatible and reuse code base
  - But must evolve the current interfaces

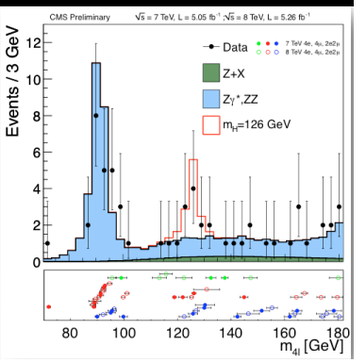
- Gradual introduction of new ***backward incompatible*** interfaces in a 'new' namespace:

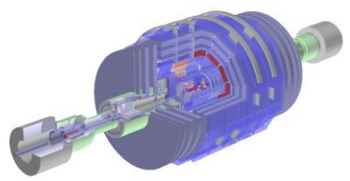
```
ROOT::T...
aliased with
ROOT::v7::T...
```

"Things alter for the worse spontaneously, if they be not altered for the better designedly."
Francis Bacon

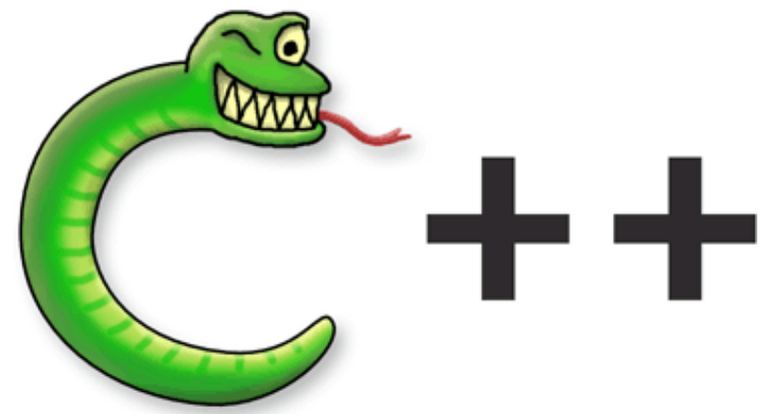
Collaborations





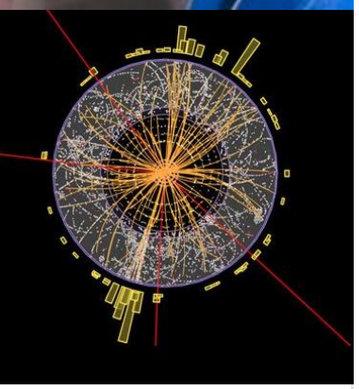


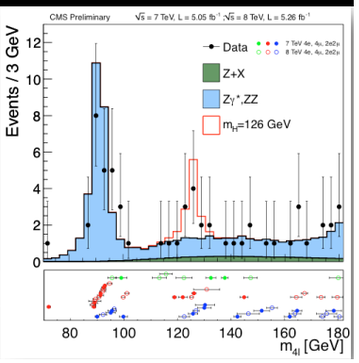
- Effort on ***I/O*** have been shared/decentralized in the last few years, could expand some more.
- ***Python*** (see Wim's poster)
- Math
  - ***R*** package (both directions)
  - New Random generators for concurrent environment
    - work within ***MixMax*** project funded by the ***EU***

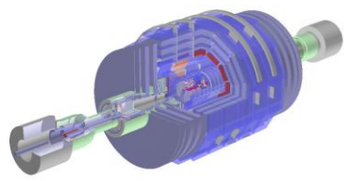


- Refresh "***How To Contribute***" page
  - Bring up to date for ***git***, ***C++11***, etc.
  - Explicit list of outstanding projects
- Better, more flexible integration both up & down
  - Dependencies: ***VecGeom***, ***VecMath***
  - Steps towards ***BOOT*** (modules brought on demand)
  - User Projects
  - Can ***HSF*** help here?

Conclusion

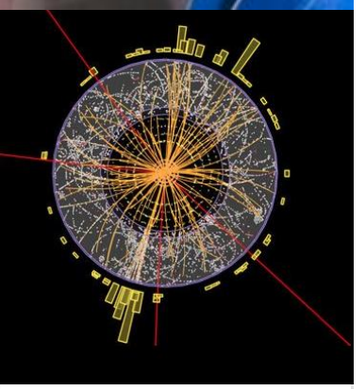


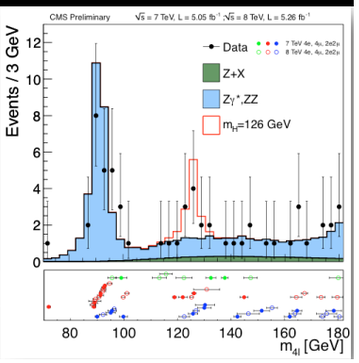


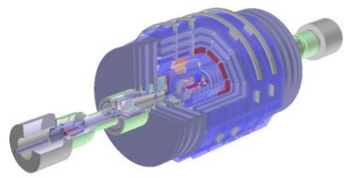


- ***ROOT*** Modernization underway
  - Starting to add ***new*** API that will overtime replace then deprecated historical API
  - Making writing [physics [analysis]] code even simpler, more intuitive and more robust

- Main Driving Principles
  - Simplicity
  - Robustness
  - Performance
    - Embrace multi-tasking and vectorization
  - Provide even better features
  - Continue our many collaborations (e.g. ***Python***, ***R***, ***I/O***)







- **ROOT-Turns-20 Users' Workshop**

- 15-18 Sept 2015 in Saas-Fee, Switzerland.
- Topics include
- Users' feedback
- ROOT Development
- Impressions from the outside
  - Bjarne Stroustrup, the inventor of C++ (TBC)
  - EA Sports, on how they do large-scale data analysis for instance for game data (TBC)
  - Data analysis with other products

Backup Slides

Release schedule

- ***ROOT*** 6.04 release on Tuesday, May 26
  - MCJIT

- v5.34 patches branch returns to bug fix only mode
  - Releases on demand

- 6 months releases cycle (v6)

ROOT Main Directions

- ***Cling*** Interpreter and its full exploitation
  - ***C++11/14***, JIT compilation opens many possibilities
  - e.g. *TFormula*, automatic differentiation, etc.

- Parallelization
  - Seek any opportunity to better exploit new(ish) hardware

- Performances improvements
  - Amdahl, File Format, Streaming, Vectorization

- Re-thinking interfaces; Simplification and Clarification
  - Leverage ***C++11*** for ease of use/documentation
  - Explore new ways to present and access data (eg.thin clients)

Here comes cling

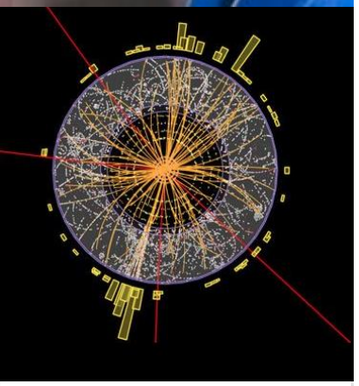
- ***Cling*** introduces binary compatible Just In Time compilation of script and code snippets.

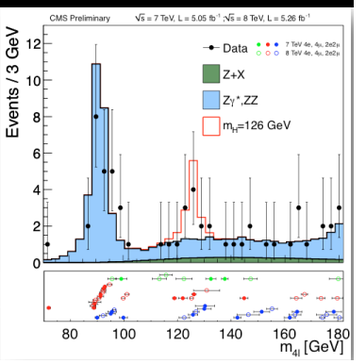
- Will allow:
  - ***I/O*** for 'interpreted' classes
  - Runtime generation of ***CollectionProxy***
  - Run-time compilation of ***I/O*** Customization rules
    - including those carried in ***ROOT*** file.
  - Faster, smarter ***TTreeFormula***
  - Potential performance enhancement of ***I/O***
    - Optimize hotspot by generating/compiling new code on demand
  - Interface simplification thanks to full ***C++*** support

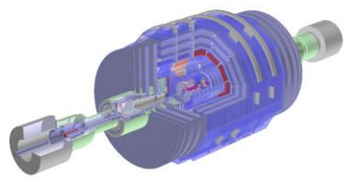
File Format Upgrades

- Switch to little-endian
  - Enable additional run-time optimization
- Support ***C++11*** entities
- Improve meta-data
  - Reduce cost of repeated [deep] hierarchies
- Space saving changes.
  - Improve compression of branch of unsplit collections
  - Reduce overhead for deep hierarchy
- Time saving changes
  - Compress each entry individually to improve random access
- Write-once files
  - Support for direct write to ***Hadoop*** file System
- ***SQLite*** within ***ROOT*** file
  - Support database (for meta-data) co-located with data

Histograms

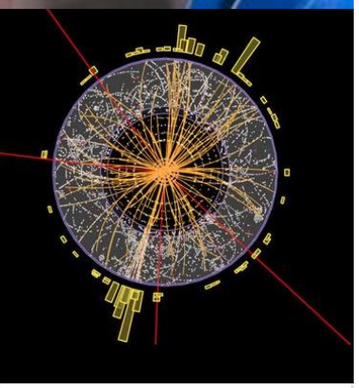


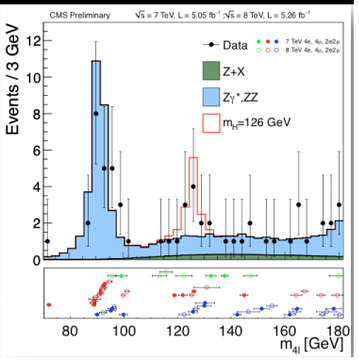


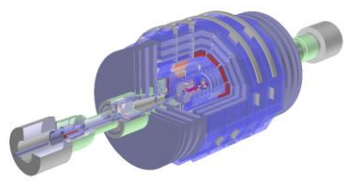


- TFormula/TF1 rewritten to leverage cling
  - < …. Words from releases notes …. >
- Histograms
  - TH1::kCanRebin -> TH1::SetCanExtend(TH1::kXaxis)
  - TH1::Sumw2() behavior now default for histogram filled with weights different than one.

Math

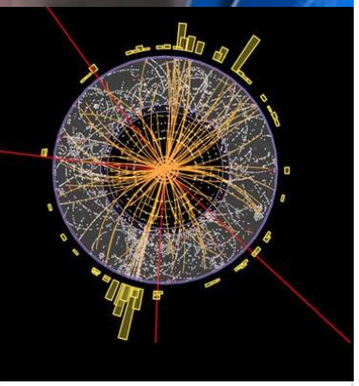


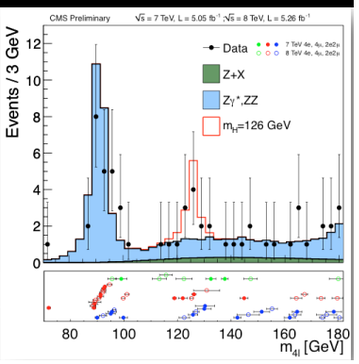


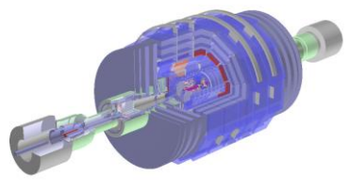


- Implement parallelisation where needed
  - e.g. fitting, profile likelihood scans, grid searches, etc..
- Improve MVA tools in ROOT
  - add some new algorithm (e.g. variable importance, multi-target regression)
  - add interface for R to use MVA tools of R in TMVA
  - investigate and replace (if needed) some of the tools
  - improve kd-tree's to use for interpolation and density estimation in multi-dimensions.
- RooFit
  - improve performances looking a real case-models (e.g. Higgs combination models)
  - exploit vectorization in pdf evaluations in RooFit/HistFactory
- RooStats
  - extend support for 2D models in RooStats
  - facilitate usage of tools (e.g. command line for running RooStats limit and significance tools)
- New Random generators for concurrent environment
  - work within MixMax project funded by the EU

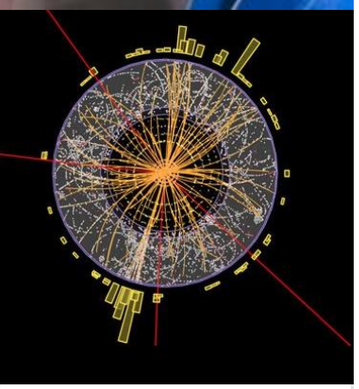
Distribution

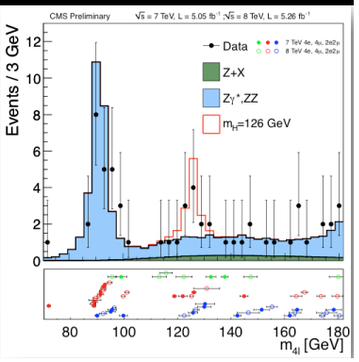


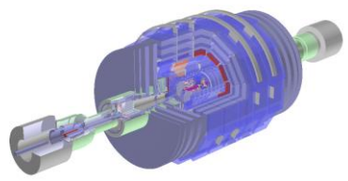




- Auto-install (and compile as needed) from a small core?
- To Host or not to host, that is the question.
- Migrate TGeom to use VecGeom
  - keep the same user interface if possible
- Distribute rootpy within ROOT
  - The rootpy project is a community-driven initiative aiming to provide a more pythonic interface with ROOT on top of the existing PyROOT bindings
- Interesting Q in regards to HSF.




CMS Preliminary
√s = 7 TeV, L = 5.05 fb⁻¹ ; √s = 8 TeV, L = 5.26 fb⁻¹
Data
Z+X
Zγ*,ZZ
m_H=126 GeV
Events / 3 GeV
m_4l [GeV]




YES!