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
• **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

• **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
• **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.
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
  \[ \exp(-0.5 \times (x-1)/2)^2 \]
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

• 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

**TTreeReader**

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

No SetBranchAddress or GetEntry

• Introduced **TTreeReader**:
  • 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

\[ \prod_{j \geq 2} \left( \sum_{k \geq 0} a_{jk} z^k \right) = \sum_{n \geq 2} z^n \left( \sum_{k_1, \ldots, k_n} a_{k_1} a_{k_2} \cdots \right) \]

\[ W_{\delta_1 \delta_2 \sigma_1 \sigma_2} = U_{\delta_1 \delta_2}^{3d} + \frac{1}{8 \pi} \int_{\mathbb{R}^4} d^4 x \left[ \frac{U_{\delta_1 \delta_2}^{3d} - U_{\delta_1 \delta_2}^{3d}}{U_{\delta_1 \delta_2}^{3d}} \right] \]

\[ dr = \frac{1}{2 \lambda} \left( \prod_{j \geq 2} \left( \frac{1}{(2 \pi)^2} \right) \right) \lambda^2 (m_A - (p_i))^2 (2n! 4^d (p_A - \Sigma p_i)) \]

\[ 4 \Re \left\{ \frac{2}{2} \chi(s) \left[ g_1^s g_1^a (1 + \cos^2 \theta) + g_1^a g_1^a \cos \theta \right] \right\} \]

\[ p(n) = \frac{1}{n \sqrt{2}} \sum_{k=1}^{\infty} \sqrt{k} A_k(n) e^{-\frac{1}{2} \cosh \left( \frac{\sqrt{n-1}}{\sqrt{n-2}} \right)} \]

\[ \left( \frac{v+1}{2} \right)^{\frac{C^TE}{2n}} N \subset \mathbb{R} \]

RHIC スピン物理 ニューヨーク

Option CANDLE example

Option VIOLIN example
Say No To Rainbow Palette

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

  ```
  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

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

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

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

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

```cpp
// 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 \texttt{C++11} constructs
  – \texttt{std::string}, \texttt{std::string\_view}
  – \texttt{std::array}, \texttt{std::shared\_ptr}, \texttt{std::unique\_ptr}
  – new \textit{STL} collections.

• Continue to explore, standardize and offer \textit{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**)

Philippe CANAL
root.cern.ch

CHEP 2015 - Okinawa 2015
13 April
• **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.