ROOT AND NEW PROGRAMMING PARADIGMS

PHILIPPE CANAL, FERMILAB
AXEL NAUMANN, CERN

FOR THE ROOT TEAM

2016-10-10, CHEP 2016 / SAN FRANCISCO

2 A LONG LONG TIME AGO IN A DISTANT LAND …

- Early days of C++
  - No class templates or function templates
  - No standard containers
  - No standard smart pointer
  - No practical exception handling (significant performance penalty)
  - No threading model, no filesystem abstraction
  - No move semantic, no initializer-lists
  - Not even a standard string
  - OO and virtual functions on the high of hype
- So ROOT
  - Provided Collections, Strings
    - e.g TClonesArray implemented 'emplace_back' 15 years before it appeared in the standard.
  - Had to rely on polymorphism
  - To implement shared ownership had to rely on (implicit) global registries
  - TThread, TSystem
  - Interface relying heavily on pointer and statics
  - Error handling via nullptr and error/warning message handler.

3 C++ NOWADAYS

- C++11 own goals
- Improve Robustness
- Improve Speed
- Improve Code clarity

- Standard libraries with very efficient collections, smart pointers
- Fast exception handling
- Threading model (and a filesystem abstraction in C++17)
- Move semantic, Initializer-list
- Efficient templates, template alias, perfect forwarding
- Reliable static analysis tools
- And a standard string …

4 C++ IS STILL EVOLVING

- From Stroustrup: "Within C++ is a smaller, simpler, safer language struggling to get out"
  - Code can be simpler
  - as efficient as ever
  - as expressive as ever
- Based on a combination of
  - Using standard features
  - Zero cost abstraction (gsl::owner<T>)
  - Static analysis tool to verify that those features and abstractions are used consistently
- See Stroustrup's ROOT Workshop presentation "Writing Good C++14" filmed at CppCon and "The Evolution of C++ Past, Present and Future"

5 BACKWARD COMPATIBILITY

- For 20 years now, ROOT macros "just" worked across ROOT versions:
  - TFile* f = new TFile("hist.root");
  - hpx->Draw();

That's Good, Right?

- Dated interface personality
- Functionality *changes* impossible

6 TYPICAL ROOT SCRIPT, SHORT AND SIMPLE, ISN'T

```
TFile *f = TFile::Open(fname,"NEW");
TH2D *h1 = new
  TH2D(name1,title1,100,10,20,50,-5,5);
double binsy[5] = {0., 1., 5., 10., 50.};
TH2D *h2 = new
  TH2D(name2,title2,50,10,20,4,binsy);
for(long i = 0; i < kN ; ++i)
{
    h1->Fill(x[i],y[i]);
    h2->Fill(x[i],y[i]);
}
f->Write();
delete h2;
delete h1;
delete f;
```

7 BUT WHAT ABOUT THIS VARIATION?

```
TFile *f = TFile::Open(fname,"NEW");
TH2D *h1 = new
  TH2D(name1,title1,100,10,20,50,-5,5);
double binsy[5] = {0., 1., 5., 10., 50.};
TH2D *h2 = new
  TH2D(name2,title2,50,10,20,4,binsy);
for(long i = 0; i < kN ; ++i)
{
    h1->Fill(x[i],y[i]);
    h2->Fill(x[i],y[i]);
}
f->Write();
delete h2;
delete h1;
delete f;
```

```
TH1D *h1 = new
  TH1D(name1,title1,10.0,20.0,50);
TFile *f = TFile::Open(fname,"Recrate");
double binsy[5] = {0., 1., 5., 10., 50.};
TH2D *h2 = new
  TH2D(name2,title2,50,10,20,5,binsy);
for(long i = 0; i < kN ; ++i)
{
    h1->Fill(x[i],y[i]);
    h2->Fill(x[i],y[i]);
}
f->Write();
delete f;
delete h1;
delete h2;
```

8 BUT WHAT ABOUT THIS VARIATION?

```
TFile *f = TFile::Open(fname,"NEW");
TH2D *h1 = new
  TH2D(name1,title1,100,10,20,50,-5,5);
double binsy[5] = {0., 1., 5., 10., 50.};
TH2D *h2 = new
  TH2D(name2,title2,50,10,20,4,binsy);
for(long i = 0; i < kN ; ++i)
{
    h1->Fill(x[i],y[i]);
    h2->Fill(x[i],y[i]);
}
f->Write();
delete h2;
delete h1;
delete f;
```

```
TH1D *h1 = new
  TH1D(name1,title1,10.0,20.0,50);
TFile *f = TFile::Open(fname,"Recrate");
double binsy[5] = {0., 1., 5., 10., 50.};
TH2D *h2 = new
  TH2D(name2,title2,50,10,20,5,binsy);
for(long i = 0; i < kN ; ++i)
{
    h1->Fill(x[i],y[i]);
    h2->Fill(x[i],y[i]);
}
f->Write();
delete f;
delete h1;
delete h2;
```

Can you spot 6 mistakes?
The compiler can NOT!

9 DETAILS ON THE MISTAKES

- Storing only one of the two histograms
- 1D histo has the wrong number of bins and wrong range.
- File opened read-only
- Not enough bin border (“bins out of order”)
- 1D histo filled with y-coord as weight
- double delete

```
TH1D *h1 = new
  TH1D(name1,title1,10.0,20.0,100);
TFile *f = TFile::Open(fname,"Recrate");
double binsy[5] = {0., 1., 5., 10., 50.};
TH2D *h2 = new
  TH2D(name2,title2,50,10,20,5,binsy);
for(long i = 0; i < kN ; ++i) {
    h1->Fill(x[i],y[i]);
    h2->Fill(x[i],y[i]);
}
f->Write();
delete f;
delete h1;
delete h2;
```

10 BEHIND THE SCENE

```
TFile *f = TFile::Open(fname,"NEW");
TH2D *h1 = new
  TH2D(name1,title1,100,10,20,50,-5,5);
double binsy[5] = {0., 1., 5., 10., 50.};
TH2D *h2 = new
  TH2D(name2,title2,50,10,20,4,binsy);
for(long i = 0; i < kN ; ++i)
{
    h1->Fill(x[i],y[i]);
    h2->Fill(x[i],y[i]);
}
f->Write();
delete h2;
delete h1;
delete f;
```

Global variable gDirectory used in TH2D constructor

Global list of objects interested in being told about *every* object deletions (gROOT->GetListOfCleanups()).

Flat long list of arguments:

```
TH2D(const char*,const char*,Int_t,Double_t,Double_t,
Int_t,Double_t ,Double_t);
[Compiler can not check semantic of variable name]

 Int_t Fill(Double_t x,Double_t y);
 Int_t Fill(Double_t x,Double_t y,Double_t w);
```

11 INV7 ....

```
TH2D h1({100, 10.0, 20.0},
        { 50, -5.0,  5.0});
TH2D h2({50, 10.0, 20.0},
        {{0., 1., 5., 10., 50.}});
for(long i = 0; i < kN ; ++i)
{
    h1.Fill({x[i],y[i]});
    h2.Fill({x[i],y[i]});
}
TFilePtr file = TFile::Create(fname);
file->Write( name1, h1 );
file->Write( name2, h2 );
```

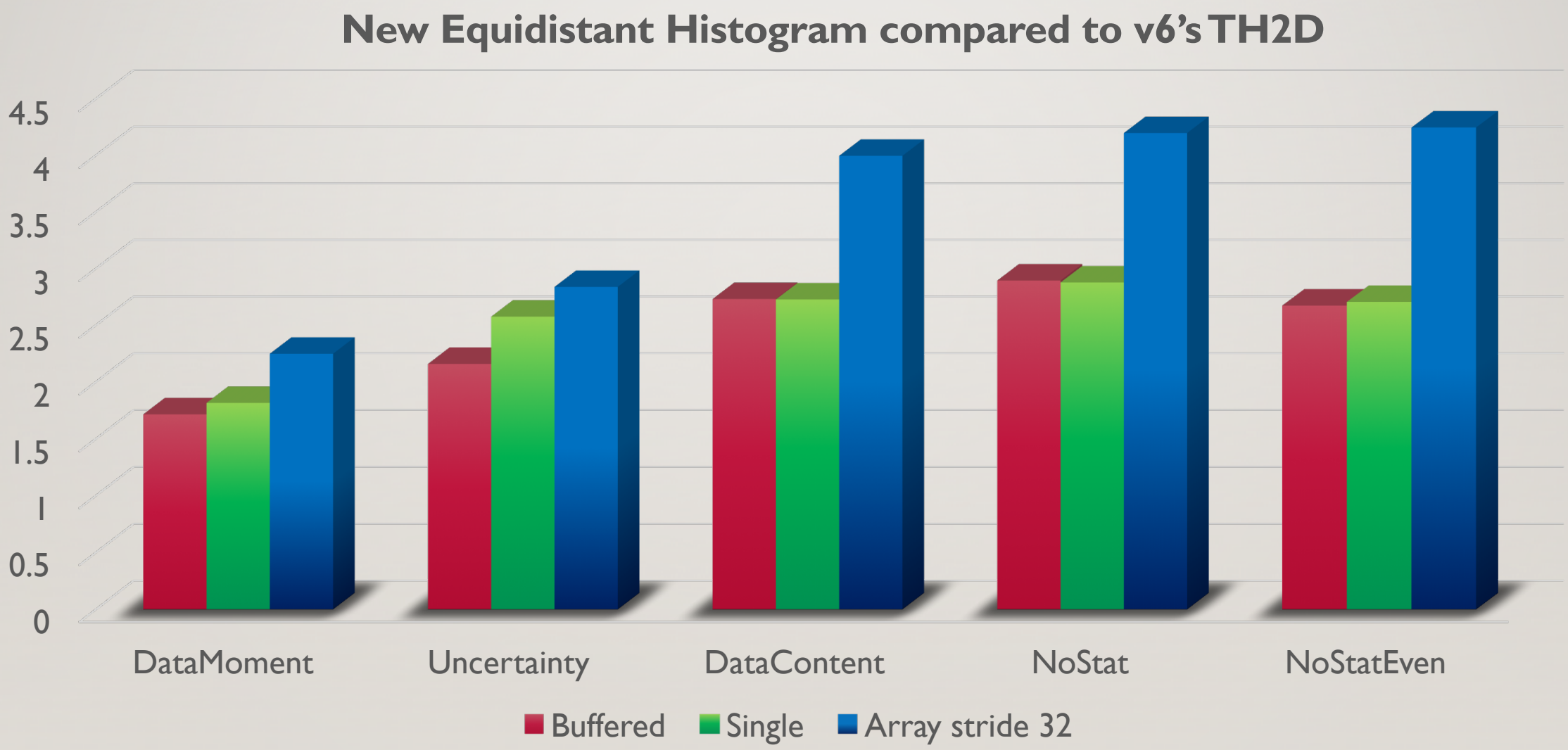
12 IN V7 …. FAILURE CAUGHT/PREVENTED BY THE COMPILER

```
TH1D h1({10.0,20.0,50});
TH2D h2({50, 10.0, 20.0},
        {{0., 1., 5., 10., 50.}});
for(long i = 0; i < kN ; ++i)
{
    h1.Fill({x[i],y[i]});
    h2.Fill({x[i],y[i]});
}
TFilePtr file = TFile::Create(fname);
file->Write( name1, h1 );
file->Write( name2, h2 );
```

```
error: type 'float' cannot be narrowed
to 'int' in initializer list
… TH1D h1({10.0,20.0,50})
            ^~~~
```

```
error: excess elements in struct
initializer
h1.Fill({x[i],y[i]});
              ^
```

13 BETTER SYNTAX + BETTER PERFORMANCE



14 NEW INTERFACES FOR ROOT

Goals:

- Simplicity, Robustness
  - Clear ownership, type-safety, compiler-checkable options
- Interoperability, Task Parallelism
- Design for change: abstraction, separation of public and implementation details
- Improve speed where possible
- Be used in experiment code developed for CERN Run 3

Means:

- Current C++
  - including smart pointers
- Reduce global state
  - Make any global state alteration explicit

```
TFilePtr f =
   TFile::OpenForRead("hist.root");
auto hist = f->Get<TH1F>("hpx");
canv->Draw(hist);
```

15 THE PLAN

- Gradual Transition
  - New prototype interfaces in ROOT::Experimental and header ending in .hxx
  - Released one by one by moving into ROOT::
  - Glue new interfaces to rest of ROOT 6
  - Later: ROOT 6 interfaces use ROOT 7 ones
- Design with care, take time: these interfaces should survive for the next 20 years!
- Allow reading old data into new ROOT
- Interact with YOU, early and continuously
  - take what worked well for ROOT 6

16 CURRENT STATE

- TFile interface
  - Name now part of the connection with container rather than within the object
- Histogram including read/write to file.
  - Can customize what kind of statistics is being gathered ***without*** any run-time penalty
- Starting with TPad/TCanvas
  - On going discussion on how to express Draw options such that it is intuitive, flexible and compile time checked while remaining easy to code and read.

17 CONCLUSION

- The world (and C++) has changed, ROOT needs to adapt
- Successful maintenance, yet need for evolution
- Can only convince through features, robustness, simplicity: usability
- Small steps enable organic growth and has enabled feed-back loop

- **V7 needs YOU**
- both physicists and computing aficionados
- Get involved by signing up at https://cern.ch/root7-signup for mailing list and semi-regular Wednesday meeting!
- Discussions have already produced better interfaces and implementation and lead to contribution of patches and pull requests!