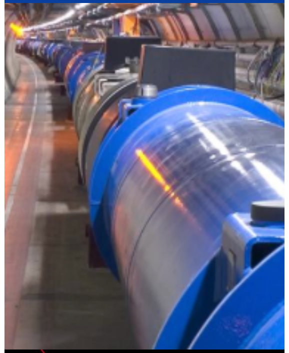
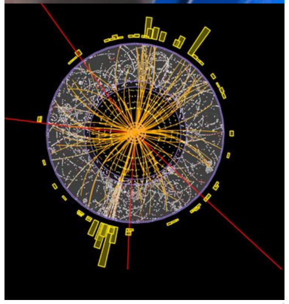
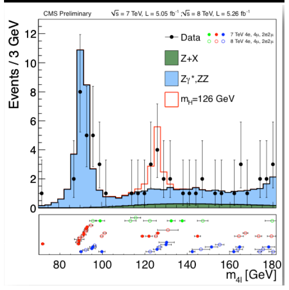
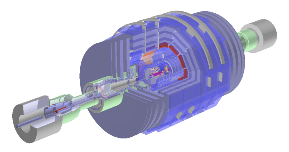
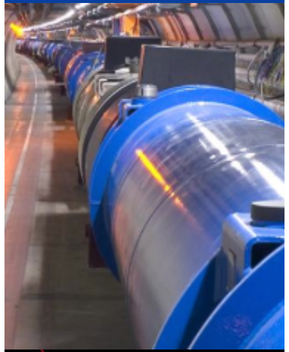
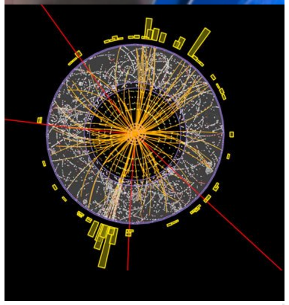
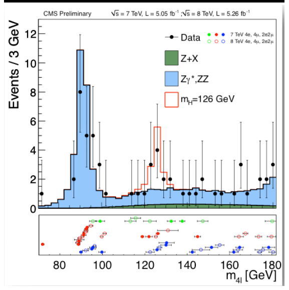




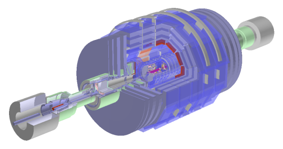


ROOT I/O Workshop
May 25th 2016

Philippe Canal
Fermilab
On behalf of ROOT Team.

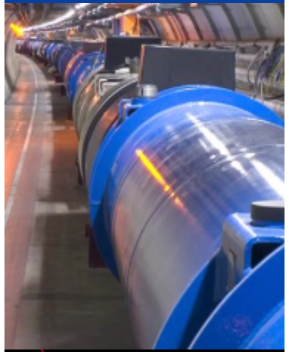
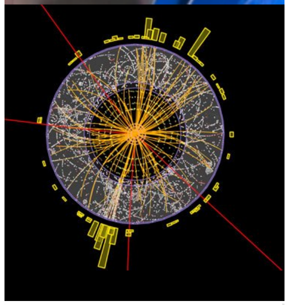
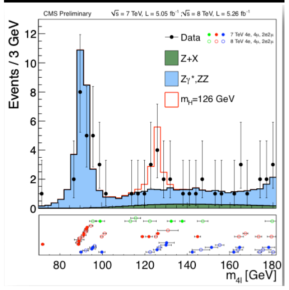
Efforts

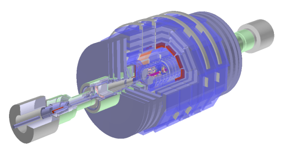






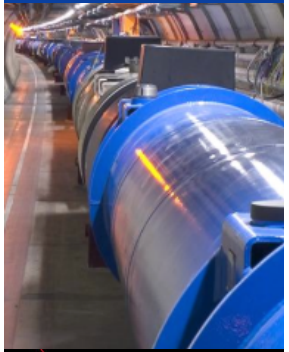
- Slices of Danilo
- Slivers of Philippe
- Zhe Zhang between classes
- Brian squeezing through grids
- David S. until back drafted
- Axel around a coffee
- Enric lock picking
- Akos Hajdu reading tree leaves
- Oliver Freyermuth githubing
- Mattias Ellert sweeping plugins
- Christian Pulvermacher pushing TClonesArray
- Oliver documenting.
- And the ever elusive Nebraskan

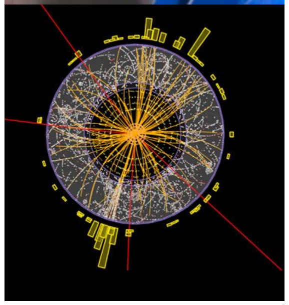
Recent additions

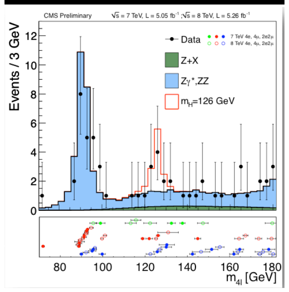




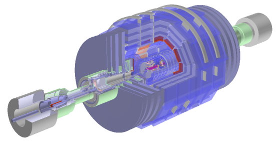


- std::unique_ptr
- Memory leaks and valgrind cleanup
- TTreeReader default for MakeSelector.
- TTree Caching in fast merging (hadd)
  - ToHumanReadableSize and FromHumanReadableSize for hadd
- Thread safety enhancements in Core, I/O, TTree
- Cling v6 migration
- gcc 5/6 and **rootStaticAnalyzer** induced code cleanups
- Prototypes of parallel reading/writing of Ttree and/or executing selectors.









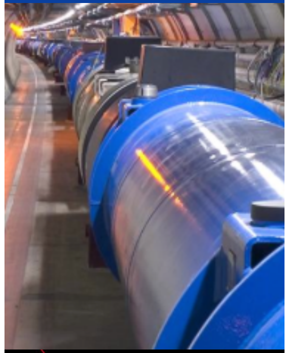
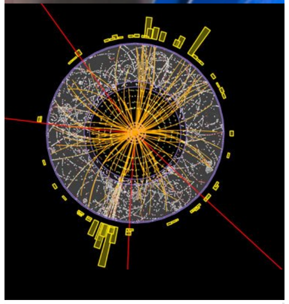
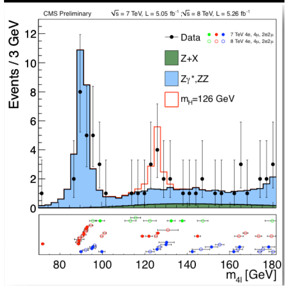
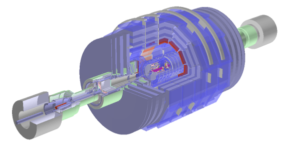
- **rootStaticAnalyzer**
  - A not-so-static post-compile-time analyzer for ROOT and ROOT-based projects.

- Implemented tests:
  - Construction/Destruction
  - Working IsA
  - Unstreamed datamembers from base-classes
  - Simple streaming after default construction
  - Test for streaming of uninitialized data after default construction

Developed and maintained by Oliver Freyermuth

See https://github.com/olifre/rootStaticAnalyzer

Progress on

Thanks to Zhe Zhang

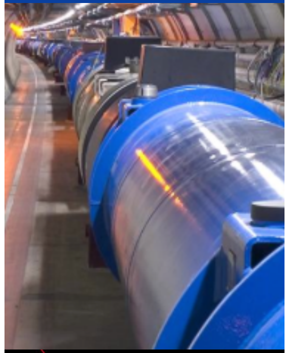
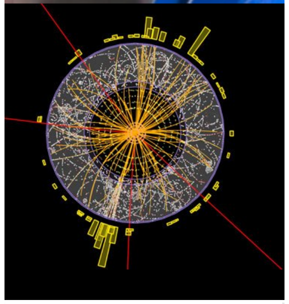
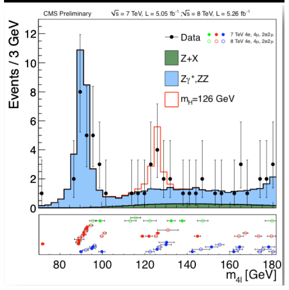
- Zero copy I/O
  - Essentially for significant performance boost
  - Prototype of TBuffer using little endian
  - First step in many
    - Extend to file. Handle/detect when I/O actions can be merged (deal with alignment etc.). Implement I/O action for the writing side.

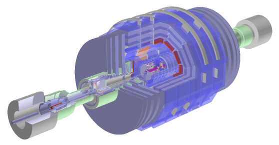
- Compress each entry individually to improve random access

- Thread safe segfault handler

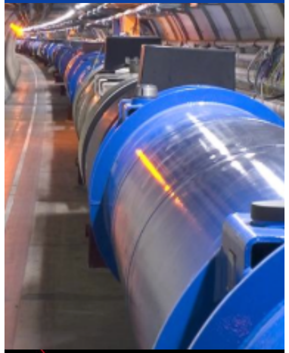
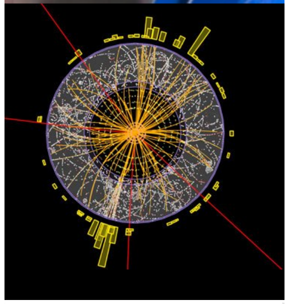
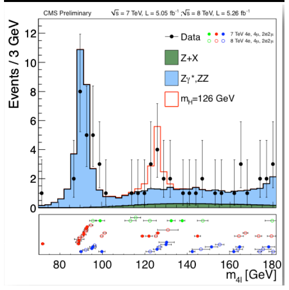
Not Yet

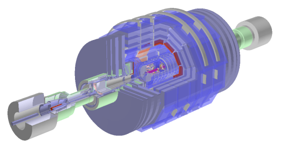
- std:array, std::shared_ptr
- ***OptimizeBasket***
  - There are a couple of new algorithm proposals
  - Need to be tested on wide range of cases
- ***TTreeCache:*** Allow alternative algorithm
- Read/WriteBuffer
  - 25% of the read code moved to optimized framework (function based) ; representing most of the use cases.
  - Write code still need to be similarly optimized



Further Plans



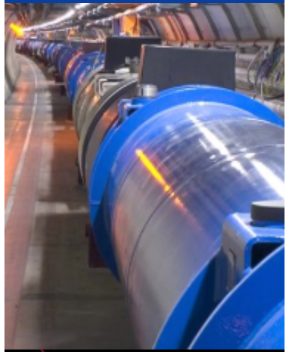
- Improve meta-data
  - Reduce cost of repeated [deep] hierarchies
  - Improve compression of branch of unsplit collections
  - Reduce overhead for deep hierarchy

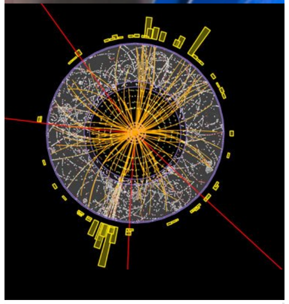
- Write one direction files (for ***Hadoop***)
- Enable Just-In-Time compilation of rules
- Extend automatic conversions
  - ***Derived***** <-> ***Base*****
  - From object to pointer
  - From ROOT Collection to STL collection

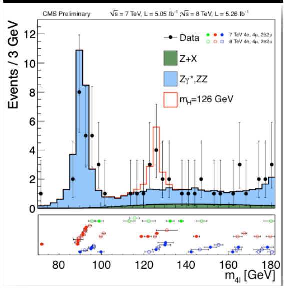
Back To The Future II

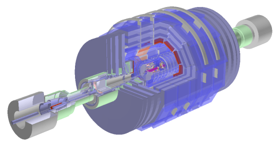
- Design proper solution for experiment requiring sliding time frames instead of events

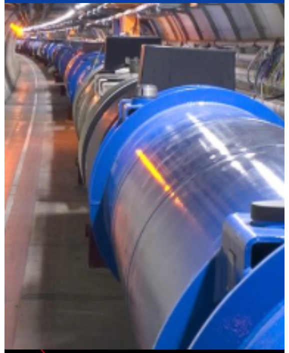
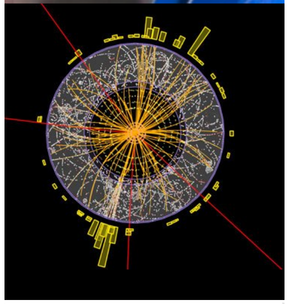
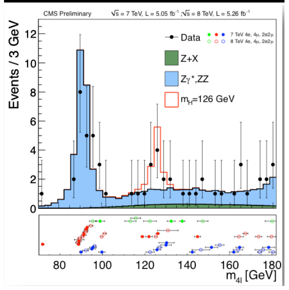
- …

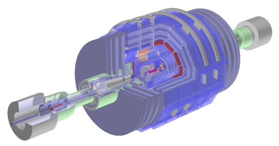




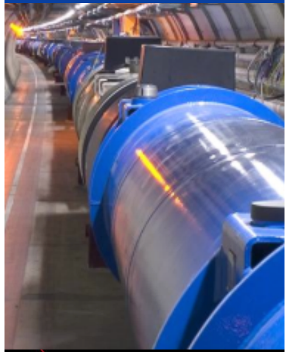
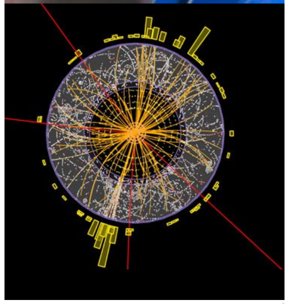
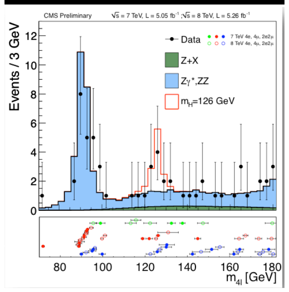


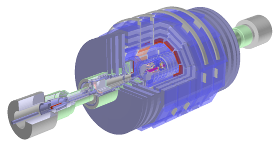




Backup Slides



- ***TTree*** Draw/Scan
  - Leverage cling
- ***TTree***
  - Interface simplification
    - Make ***SetAddress*** and ***SetBranchAddress*** 'smarter'
  - Optimizations
- In ***TTree***
  - Eg. ***TTree::Draw*** execute formula on more than one element at a time
  - New interface allowing retrieval of multiple entries at once

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