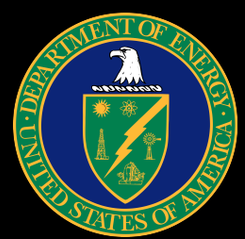
Optimizing ROOT's Performance Using C++ Modules

Dr. Vassil Vassilev
(presented by Philippe Canal)
10.10.2016

Vassil's work is entirely sponsored by USCMS and FNAL.

Fermilab



Optimizing ROOT's Performance Using C++ Modules

Recompiling C++ gets faster using C++ Modules (ISO CPP Modules TS). ROOT is on its way to use benefit from the new technology, expecting:

- faster (re-)compilation by 20-30%.
- less memory use at runtime about 40%.

ROOT's Runtime

User/Experiments' code has a lot of semantical equivalents to this.

```
// ROOT prompt (no C++ Modules):
gSystem->Load("MyLib");
// => dlopen("MyLib.so");
//    => cling->parse("1000s_of_fwd_decls.h");
MyLibClass<float> c; c.do();
// => cling->parse("#include <MyClass.h>");
```

Forces ROOT's interpreter to parse headers related to MyLib (even when we intend to use only tiny fraction of them).

This results in increased memory use and slowdown.

```
// ROOT prompt (no C++ Modules):
gSystem->Load("MyLib");
// => dlopen("MyLib.so");
//    => cling->mmap("MyLib.so.pcm");
MyLibClass<float> c; c.do();
```

C++ Modules-aware ROOT runtime will lazily allocate memory only for what you use and at the point of use!

Everything unused is mmaped.