Future of ROOT
Runtime C++ modules

Yuka Takahashi - Princeton University, CERN
Vasil Geogiev Vasilev - Princeton University
3 Very Important Summary Slides
1. With Runtime C++ Modules,

Experiments will be more Performant
2. With Runtime C++ Modules,

End users don’t have to change anything in their code
3. When you use Runtime C++ Modules,

Set `-Druntime_cxxmodules=ON` (Available in future 6.16!)
Agenda

1. C++ Modules in a nutshell
2. Effects for experiments
3. Effect for ROOT
4. Implementation
5. Status and roadmap
C++ Modules in a Nutshell
C++ Modules in a Nutshell

#include <vector>
C++ Modules in a Nutshell

#include <vector>

Textual Include
- Expensive
- Fragile

PCH
- Inseparable

Modules
- ✓
C++ Modules in a Nutshell

```cpp
#include "TVirtualPad.h"
#include <vector>
#include <set>

int main() {
...
```

original code

Compile

Parse

Textual Include

Preprocess

one big file!
C++ Modules in a Nutshell

Textual Include

1. Expensive
   Reparse the same header

2. Fragile
   Name collisions

Rcpp library

```cpp
#define PI 3.14
...
```

Users’ code

```cpp
#include <header.h>
...
double PI = 3.14;
// => double 3.14 = 3.14;
```
C++ Modules in a Nutshell

PCH (Pre Compiled Header)

1. Storing pre compiled header information (same as modules)
2. Stored in one big file
C++ Modules in a Nutshell

Modules

- Pre compiled **PCM files** contain header information
- PCMs are **separated**

Each PCM file (a.pcm) corresponds to a library (liba.so)
C++ Modules in a Nutshell

Modules

- Pre compiled **PCM files** contain header information
- PCMs are **separated**

- Compile-time scalability
- Fragility
- Separable
Effect for Experiments
Effect for Experiments

Current problem of Experiment Software Stack

- Experiments are still using **textual include**
- Not even PCH
- PCH can’t be used because it’s **too big**

Experiments are parsing **hundreds of headers** at the startup time at the moment.
Effect for Experiments

Current problem of Experiment Software Stack

- Experiments are still using **textual include**
- Not even PCH
- PCH can’t be used because it’s **too big**

Modules can do this as it’s **separable**
Effect for Experiments

Current Status

Compile ROOT in CMS environment with modules
Generate dictionary for CMS external libraries  WIP
Effect for Experiments

Current Status

Working closely with CMSSW
Getting a lot of feedback from CMS usage
  - Leads to fix bugs in Clang & Cling
Effect for Experiments

Performance Benefits

No actual benchmark yet

Simulated results in next slides
Comparison of modules to textual includes
Effect for Experiments

Memory

![Memory Chart]

- Runtime Modules
- Textual Includes

Modules

Textual Include
Effect for Experiments

CPU Time

![Bar chart showing CPU Time comparison for different modules. The chart includes categories such as 'Startup', 'hsimple', 'assembly', 'parallel_world', 'Radio-Nuclides', and 'rootgeom'. The x-axis represents the modules, and the y-axis shows CPU Time (seconds) with bars in purple for runtime modules and orange for textual includes.](chart.png)
Effect for Experiments

Experiments have hundreds of header files to parse

Thus the difference will be order of magnitude larger in experiments.
Effect for ROOT
Effect for ROOT

Correctness benefit

Without Modules

```
$ root -l
root [0] gMinuit // Cannot load variable
IncrementalExecutor::executeFunction:
symbol 'gMinuit' unresolved while linking [cling interface function]!
```
Effect for ROOT

Correctness benefit

With Modules

```
$ root -l
root [0] gMinuit // Could load libMinuit
(TMinuit *) nullptr
```
Effect for ROOT
Packaging benefit

We can make ROOT modular for lazy installing packages
- ROOT package manager
- See Oksana’s talk for more information!

Slide from Oksana :)

Yuka Takahashi 12.09.2018
Future of ROOT runtime C++ modules, ROOT Users workshop
Implementation
Implementation

User Interface

ROOT
Integration with ROOT libraries and provide ROOT specific features

rootcling
Generate pems and libraries

Cling
C++ interpreter interface

Clang
Provide fundamental implementation of C++ modules
Implementation

Clang

- External project under LLVM
- Bi-weekly meeting with C++ Modules community
- Reporting & fixing bugs
- ROOT is the largest user of Modules outside industry
Status and roadmap
Status and roadmap

Memory - hSimple

Yellow line is PCH
Green line is Modules

- Preloading all modules
- Autoloading libraries
- Check fCXXRecordDecl Cache
- Enable Cling modulemap
- Implement Bloom filter

https://rootbnch-grafana-test.cern.ch/
Status and roadmap

Memory - hSimple

Yellow line is PCH
Green line is Modules

Autoloading libraries
Implement Bloom filter
Enable Cling modulemap
Preloading all modules
Pin vtable

https://rootbnch-grafana-test.cern.ch/
Preloading Modules

All correctness benefit over PCH is due to this

- Preloading of all modules
- Replace old infrastructure
  - rootmap
Status and roadmap

Real time - hSimple

Yellow line is PCH
Green line is Modules

Hunting down bottlenecks!

https://rootbnch-grafana-test.cern.ch/
Status and roadmap

Status

Fundamental Construction in ROOT Core, which affects every code passed to ROOT
Working with industry and CMSSW
Good progress in performance optimization

Roadmap

Reach complete production level before 6.16
Continue working on optimization
Modularise CMSSW!
Thank you for your attention!
Backup Slides
Effect for ROOT

C++ Modules is a mechanism to boost **compilation time**

For ROOT, it turns into **runtime speed improvement** as we have C++ interpreter