C++ Modules in ROOT and Beyond

Vasil Georgiev Vasilev (Princeton University (US))
Yuka Takahashi (Princeton University (US))
David Lange (Princeton University (US))
Malik Shahzad Muzaffar (CERN)
Mircho Rodozov(CERN)
Oksana Shadura (University of Nebraska Lincoln (US))

Motivation

#include <vector>

Textual Include

✗ Expensive
Fragile

1. **Expensive**
   Reparse the same header
2. **Fragile**
   Name collisions

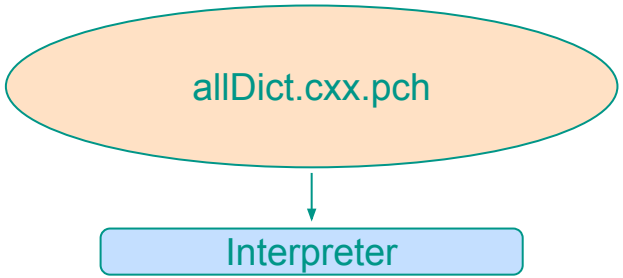
Rcpp library
#define PI 3.14
...

User Code
#include <header.h>
...
double PI = 3.14;
// => double 3.14 = 3.14;

Precompiled Headers (PCH)

✗ Inseparable

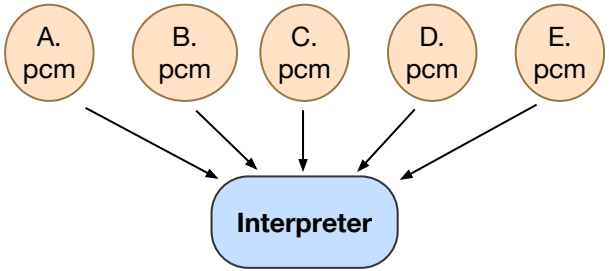
1. Storing precompiled header information (same as modules)
2. Stored in one big file
   - Inseparable



Modules

✓

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

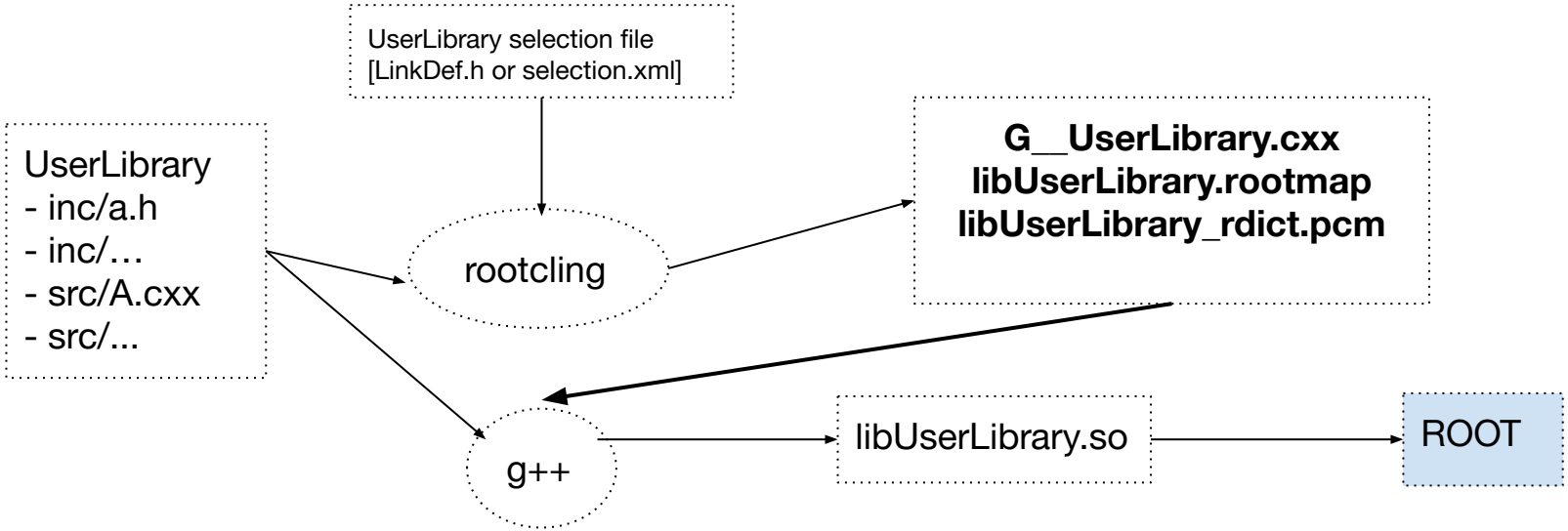


Motivation

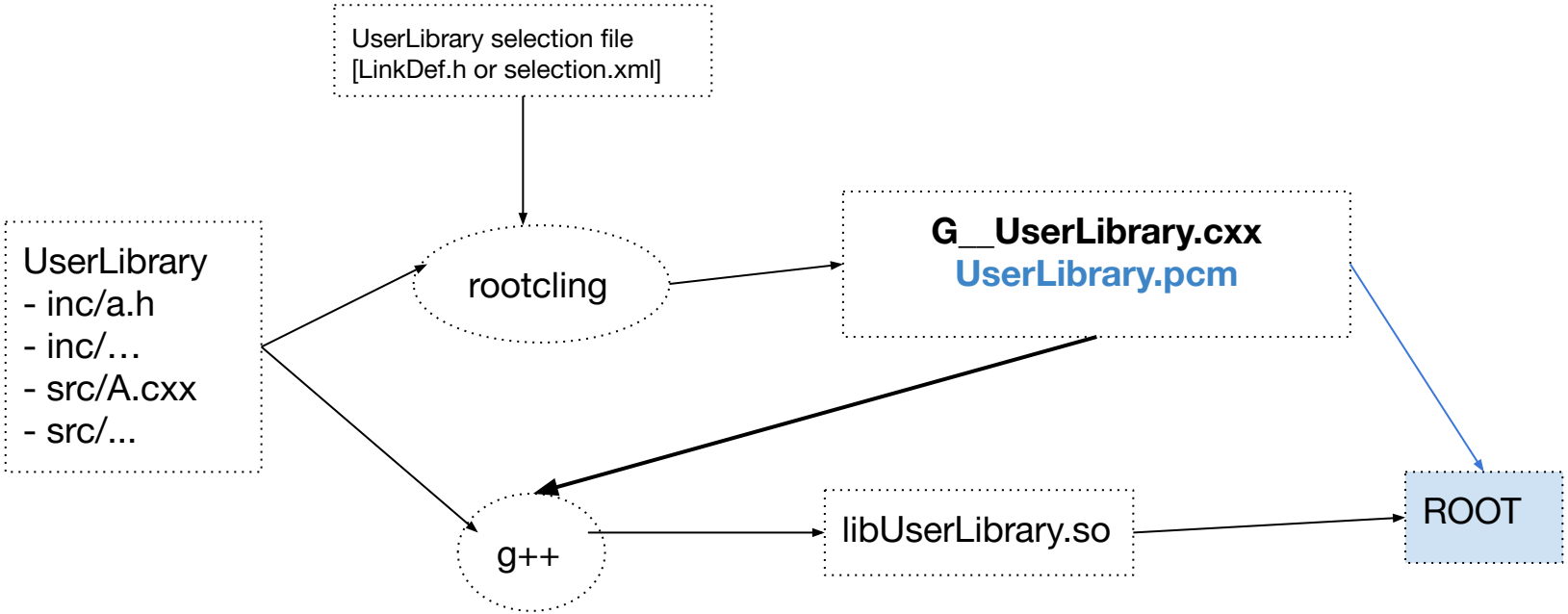
- ROOT sometimes requires parsing of sizeable chunks of C++ code to aid user actions at the prompt, dictionary setup, and various interpreter services
- **C++ Modules are designed to minimize the reparsing of the same header content** via an efficient on-disk representation of C++ Code

Dictionaries in ROOT

- ROOT provides **automatic generation of serialization information (I/O) and automatic header and dependency resolution (partially) *via dictionaries***
  - Dictionary is an extra file C++ needs to be synthesized and compiled into the defining library



Dictionaries in ROOT



Dictionaries in ROOT

G__UserLibrary.cxx has several sections which require reprocessing immutable headers in runtime

They are replaced by the much more efficient C++ Modules

```
Const char* foo = R”RAW(
-#include "TEveTrack.h"
-#include "TEveTrackEditor.h"
-#include "TEveTrackGL.h"
-#include "TEveTrackProjected.h"
-#include "TEveTrackProjectedGL.h"
-#include "TEveTrackPropagator.h"
-#include "TEveTrackPropagatorEditor.h"
-#include "TEveTriangleSet.h"
-#include "TEveTriangleSetEditor.h"
-#include "TEveTriangleSetGL.h"
...
-"TEveGedNameTextButton", payloadCode, "@",
-"TEveGeoManagerHolder", payloadCode, "@",
-"TEveGeoNode", payloadCode, "@",
-"TEveGeoNodeEditor", payloadCode, "@",
...
-class __attribute__((annotate(R"ATTRDUMP(An arbitrary polyline with fixed line and marker attributes.)ATTRDUMP"))) ..
)
```

C++ Modules in ROOT

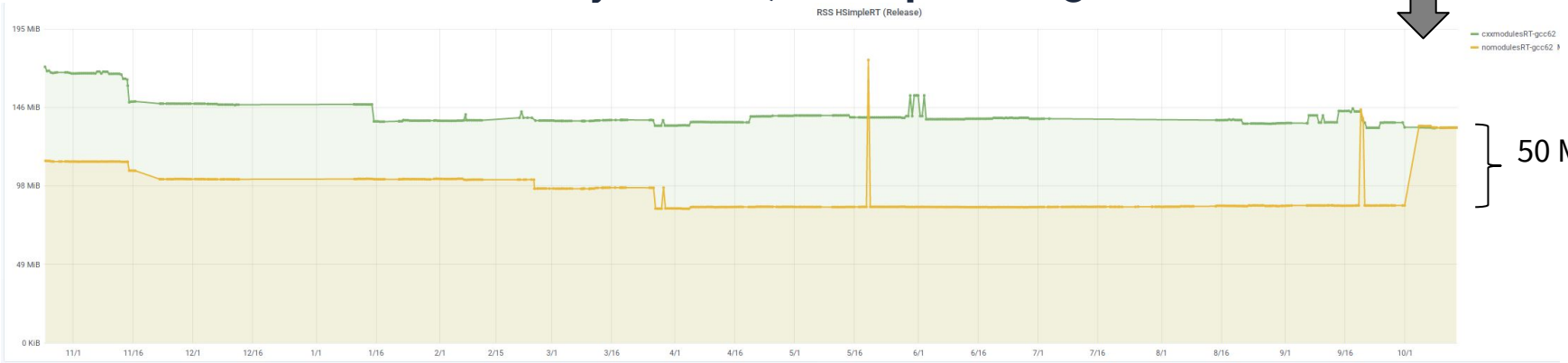
- Technology Preview released in ROOT 6.16
- **Default on UNIX in ROOT 6.20**

C++ Modules in ROOT. Loading strategies

- *Eager loading* of *pcm files at startup in ROOT 6.20
  - Small linear performance overhead depending on the number of modules
- *Delayed loading based on global module indexing (GMI)* planned for ROOT 6.22
  - **No overhead, pay only for what you use!**

C++ Modules become default on UNIX

Functionality is there, now optimizing!

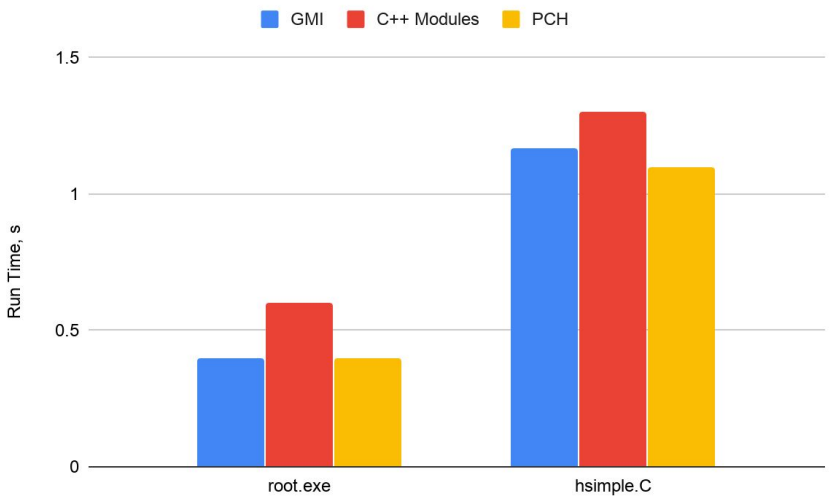


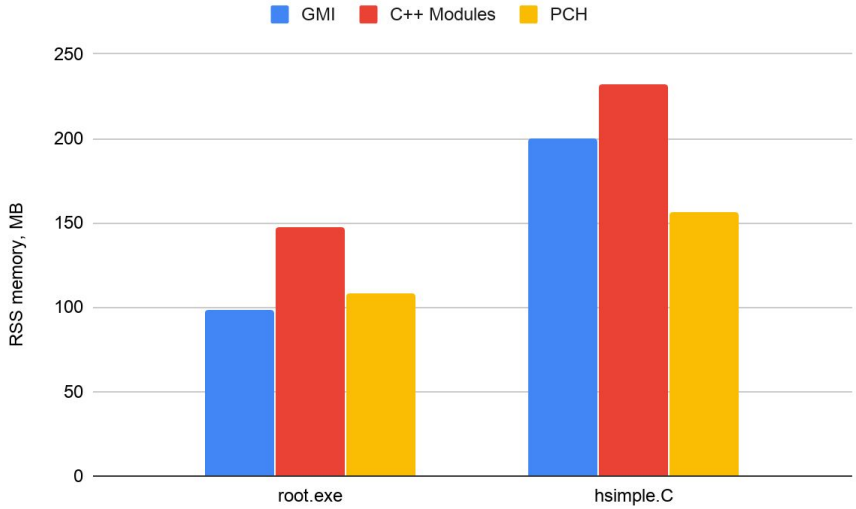
Global Module Index (GMI)

- Global Module Index (**`clang::GlobalModuleIndex`**) is **an aid for name lookup into modules**
- **Global Module Index returns a set of modules where a given identifier is present**
  - This allows us to avoid eager loading of all modules and load only necessary ones on demand

- It is using key-value store concept -> on-disk hash table
- Populated by scanning on demand all identifiers in all modules
- Created at build time (it is a on-disk file)

GMI measurements

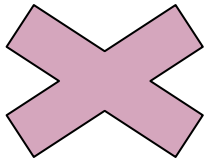


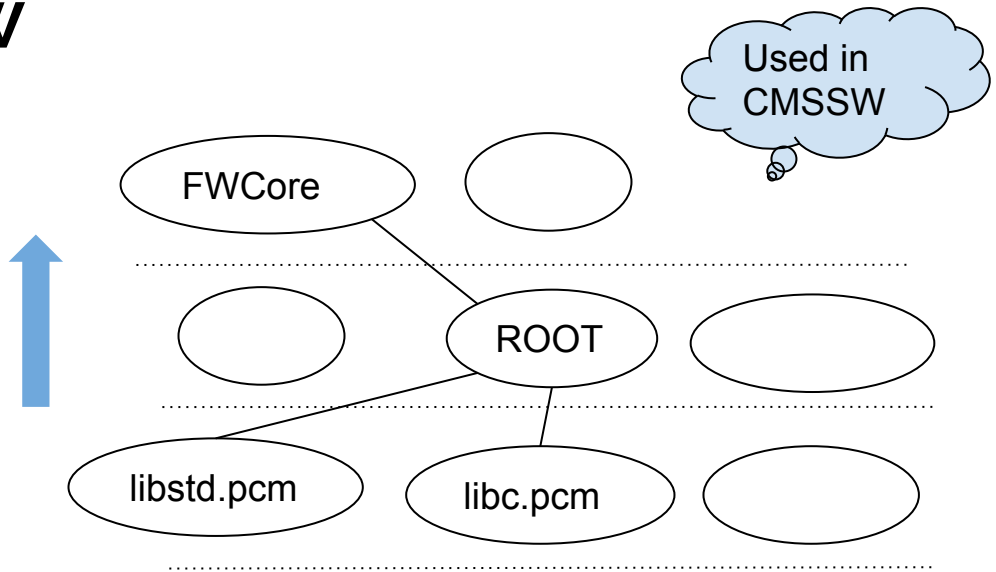


C++ Modules in CMSSW

- Available in CXXMODULE_IB

C++ Modularization in CMSSW





Top-down approach

Bottom-up approach

Library without modulemap

- Header file pollute the rest of the library dependencies
  - **The header duplication is problematic for performance but also triggers a lot of bugs when resolving the duplicate content**

Library with modulemap

- Headers are persisted in the PCM
  - **Header duplication is reduced by referencing the dependent module**

C++ Modules-aware dictionary generation in CMSSW

[module.modulemap]

- Definition file of headers to build a PCM in Clang
- Contain all “interface” headers, which are used by libraries

```
module "MathCore" {
  module “TComplex name” { header "TComplex.h" export * }
  module <name of the file> {
      header <relative path to the header file location> }
  export libMathCore.so
}
```

modulemap will contain all interface header files

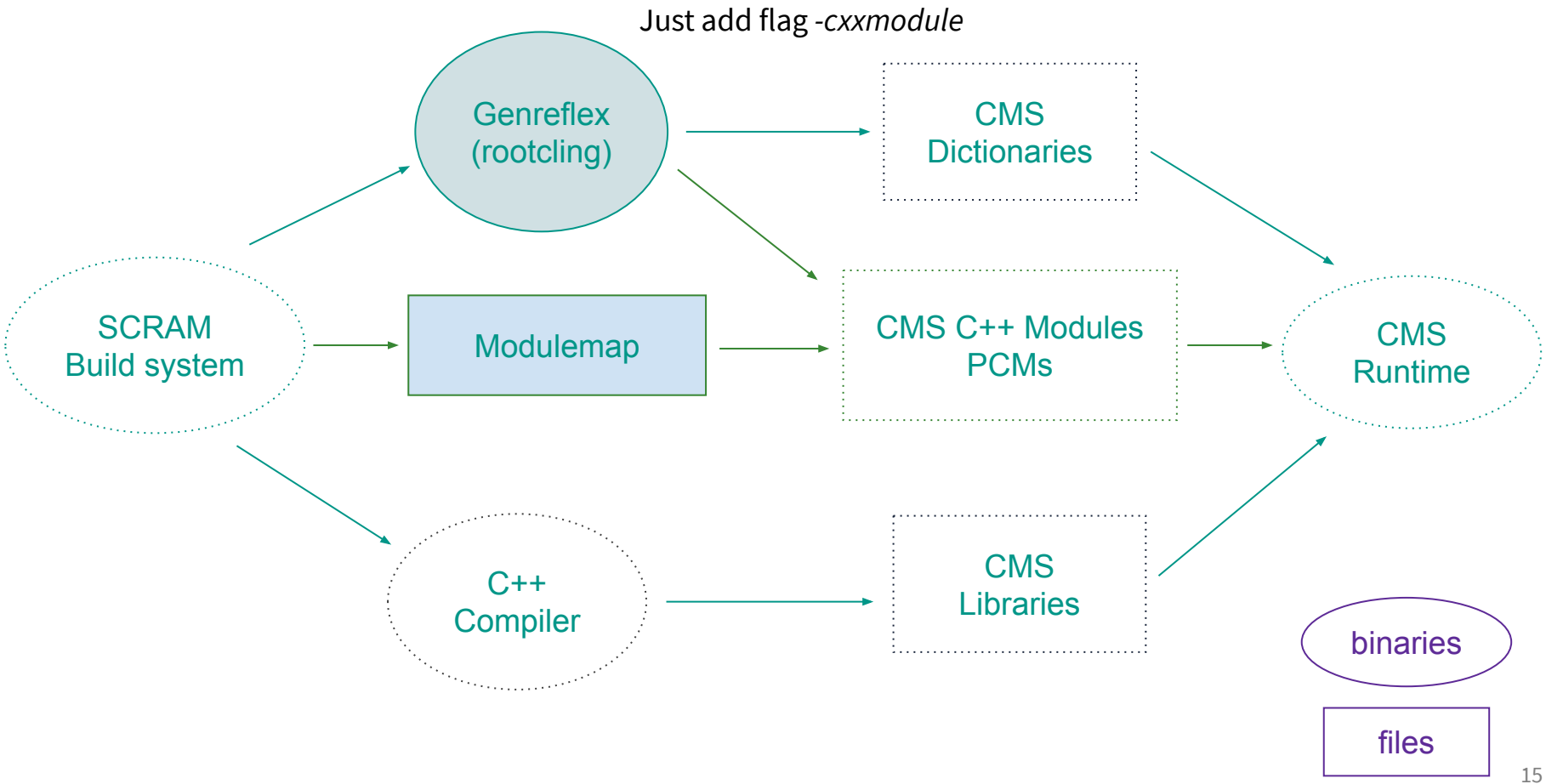
Automatic generation of modulemap

Dictionaries Using C++ Modules in CMSSW

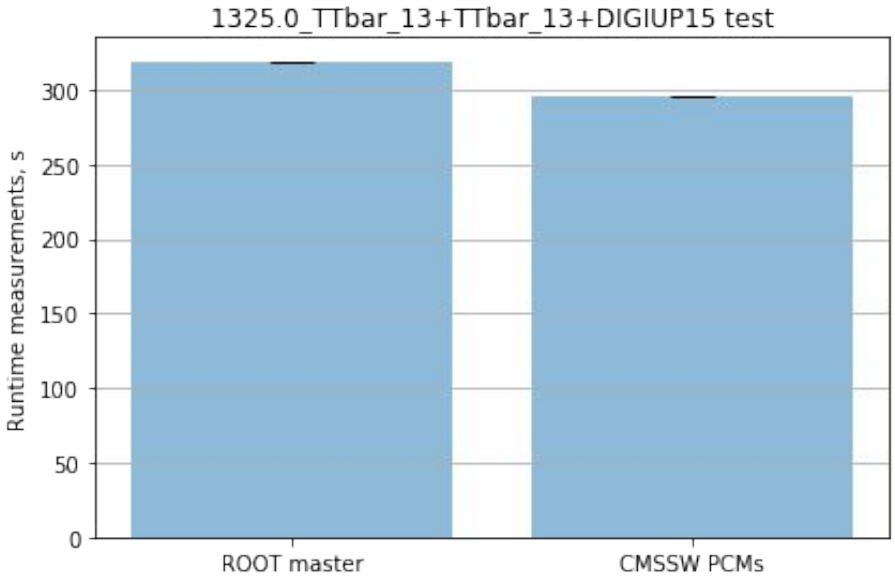
[Automatic generation of modulemap]

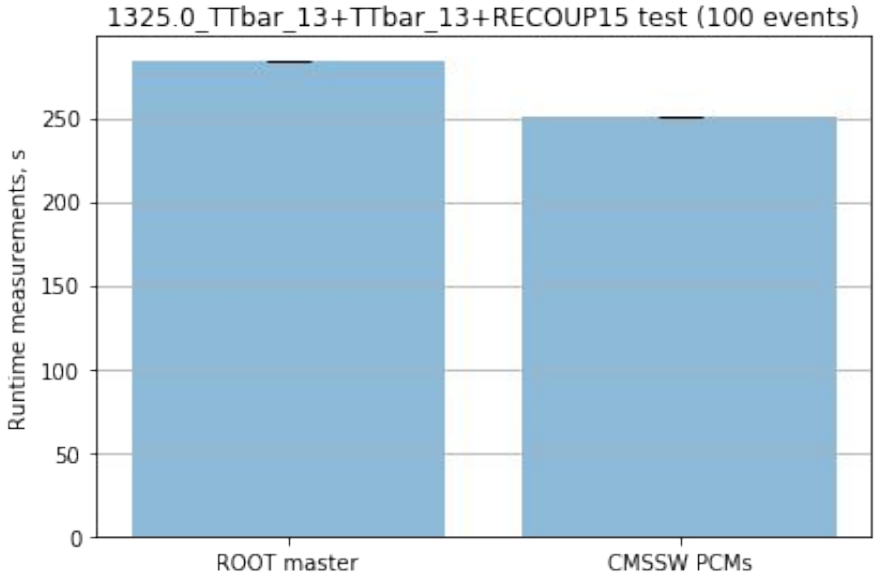
- CMSSW has “interface” headers
  - Exposed to libraries outside
- Automatically generate the modulemap by adding interface headers
  - Modulemap needs to be generated before the execution of genreflex
- *Design features*
  - *Headers can be easily vetoed from generation of modulemap via SCRAM settings*

C++ Modules-aware dictionary generation in CMSSW



C++ Modules in CMSSW. Preliminary measurements (104 modularised libraries)





The largest part of the speedup is in the initialization phase

C++ Modules in ROOT. Roadmap

C++ Modules in ROOT. Roadmap

- Modular ROOT and non-modular software stacks should work seamlessly but at no performance benefits (and costs)
- **ROOT 6.20 (January 2020):** C++ Modules will be default for all UNIX platforms (implementation based on eager module loading)
- **ROOT 6.22 (mid 2020):** C++ Modules by default for OS X (implementation based on global module indexing (GMI))

We are here to help to migrate your stack!

Thank you for your attention!

For more questions please contact: vvasilev@cern.ch
root-cxxmodules@cern.ch

Backup

Motivation of C++ Modules

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

int main() {
…
```

original code

Preprocess

Textual Include

```
                                        ……
                                        ……
# 286 "/usr/include/c++/v1/vector" 2 3
namespace std { inline namespace __1 {
template <bool> class __vector_base_commo
  __attribute__ ((__visibility__("hidden"),
__always_inline__)) __vector_base_common
                                        ……
# 394 "/usr/include/c++/v1/set" 3
namespace std {inline namespace __1 {
template <…> class set {
public:
   typedef _Key key_type;
                                        ……

int main {
                                        ……
```

TVirtualPad.h

vector

set

one big file!

Compile

Parse

.o

C++ Modules in CMSSW. Mechanism of the modulemap

[modulemap, modulemap overlay file, virtual modulemap overlay]