Optimize ROOT use of modules for large codebases

Jun Zhang GSoC 2022
Mentors: Vassil Vassilev, David Lange, Alexander Penev

Outline

- Introduction
- How modules work with ROOT
- A brief introduction to GlobalModuleIndex
- Implementation Plan
- Q&A

Introduction

C++ modules now has been adopted in ROOT by default. It greatly improved the runtime performance by reducing unnecessarily parsing a lot of huge header files. But it has its limitation.

When a user types something in the repl, ROOT will incrementally query where the identifier user input is located, or which module contains the identifier. Say we have input something like:

root [0]: edm::X

So when we have input edm, ROOT will try to load all modules that contain the identifier edm. But because edm is a NameSpaceDecl and if it contains many modules, the performance will suffer. After all, all we want is just X!

How does modules work with ROOT?

The look up logic in ROOT:

```
bool IClingCallbacks::findInGlobalModuleIndex(
    DeclarationName Name, bool loadFirstMatchOnly /*=true*/) {
  GlobalModuleIndex::FileNameHitSet FoundModules;

  // Find the modules that reference the identifier.
  // Note that this only finds top-level modules.
  if (Index->lookupIdentifier(Name.getAsString(), FoundModules)) {
    for (llvm::StringRef FileName : FoundModules) {
      StringRef ModuleName = llvm::sys::path::stem(FileName);

      // Skip to the first not-yet-loaded module.
      if (m_LoadedModuleFiles.count(FileName))
        continue;

      m_Interpreter->loadModule(ModuleName);

      m_LoadedModuleFiles[FileName] = Name;
    }
    return true;
  }
  return false;
```

What is GlobalModuleIndex

Obviously that GlobalModuleIndex is the key part of our implementation. So what exactly it is?

In short, a hashmap, that's all…

current interface:

https://github.com/root-project/root/blob/master/interpreter/llvm/src/tools/clang/include/clang/Serialization/GlobalModuleIndex.h

More details

We store the {identidier name ⇔ Module info} mapping in the llvm::OnDiskIterableChainedHashTable, which can hold the info persistently

```
122 typedef llvm::OnDiskIterableChainedHashTable<IdentifierIndexReaderTrait>
123     IdentifierIndexTable;
124
```

https://github.com/llvm/llvm-project/blob/main/llvm/include/llvm/Support/OnDiskHashTable.h

Disclamer

- New to C++ modules
- Still discovering the best solution
- Feel free to correct my mistakes

Implementation Plan

The fundamental problem is that we have lost the type info when we store the corresponding mapping, so what about just keep it?

- Solution 1: Can we just stop the lookup when we found it is a NameSpaceDecl?
- Solution 2: {DeclName ⇔ Module info} => {Decl ⇔ Module info}
- Solution 3: {DeclName ⇔ Module info} => {(DeclName, DeclKind) ⇔ Module info}

Q&A

Thanks!