Improving performance of C++ modules in Clang

Source and Production Branch

https://reviews.llvm.org/
https://llvm.org/
https://github.com/llvm/llvm-project/issues/
ROOT-patches is on top of cling-patches.

1. https://github.com/vgvassilev/clang/tree/cling-patches-llvm13
2. https://github.com/vgvassilev/clang/tree/ROOT-patches is on top of 1.

Meeting hours

https://discourse.llvm.org/t/c-modules-bi-weekly-informal-implementers-meeting/61874
https://compiler-research.org/meetings/

Machine Configuration (domestic, personal laptops) (Tapasweni)

- Mac OS big Sur
- Ubuntu 18.04
- Ubuntu 20.04
- Debian
- 1TB Hard disk, 8 GB RAM
- i5 to i7
- Clang has a perf stat system - we will use that in a first approximation.
  Notes: It is a compiler flag that you can use on your system.
  https://godbolt.org/z/s61fxoYPs

Previous work

- https://reviews.llvm.org/D41416
- Patches relatively easy to upstream:

Patch Title	Summary/Info	State	Link
Implement soft reset of the diagnostics engine.		Added to vgvassilev/**clang**.	https://github.com/vgvassilev/clang/commit/244d88da3cda561aa9b2360
Mark the file entry invalid, until reread. Invalidate SLocEntry cach readd it on reread.		Added to vgvassilev/**clang**.	https://github.com/vgvassilev/clang/commit/6ffdac0df994b96f3992b055
Propagate cache flags from LookupFile() to FileManager::getFile().		Added to vgvassilev/**clang**.	https://github.com/vgvassilev/clang/commit/0cc9535a385f0039d6ef926
Pass the OpenFile flag also to DirectoryLookup.		Added to vgvassilev/**clang**.	https://github.com/vgvassilev/clang/commit/f475cd9d1da48c1a758a20b
Survive #pragma once from virtual file.		Added to vgvassilev/**clang**.	https://github.com/vgvassilev/clang/commit/b94932abcefbaecaf4c5801
Allow interfaces to operate on in-memory buffers with no source location info.		Added to vgvassilev/**clang**.	https://github.com/vgvassilev/clang/commit/5a93d036190e2c29ec567e
Fix assertion when		Added to	https://github.com/vgvassilev/clang/commit/1c6cc386f62f9a5a87cf268b

removing decls coming from a pch/pcm		vgvassilev/**clang**.	

Problem Statement

The C++ modules technology aims to provide a scalable compilation model for the C++ language. The C++ Modules technology in Clang provides an io-efficient, on-disk representation capable to reduce build times and peak memory usage. The internal compiler state such as the abstract syntax tree (AST) is stored on disk and lazily loaded on demand. C++ Modules improve the memory footprint for interpreted C++ through the Cling C++ interpreter developed by CERN and the compiler research group at Princeton. The current implementation is pretty good at making most operations on demand.
However in a few cases, we eagerly load pieces of the AST, for example at module import time [1] and upon selecting a suitable template specialization. When selecting the template specialization we load all template specializations from the module files just to find out they are not suitable. There is a patch [2] that partially solves this issue by introducing a template argument hash and use it to look up the candidates without deserializing them. However, the data structure it uses to store the hashes leads to quadratic search which is inefficient when the number of modules becomes sufficiently large.

Test Project(s)

What do you think would be apt?

1. Size — du -hs *pcm
2. Memory Consumption — /usr/bin/time -v root.exe -l -b -q tutorials/hsimple.C
3. Use the internal performance counters in clang - https://godbolt.org/z/s61fxoYPs

Tasks

☐ Investigate and resolve eager deserialization where possible
☐ Open a review, merge the patch in llvm, revert the relevant patch from ROOT, backport the mainline patch and check if all test pass: https://github.com/root-project/root
☐ Rework the patch to use on-disk hash tables to avoid the quadratic search complexity
- https://reviews.llvm.org/D41416

☐ Develop the necessary test cases
☐ Tests, CI, Documentation.
☐ Measure performance improvements
☐ How to model the partial template specializations

Bugs To Resolve

Bug	Fixes/Usefulness	PR	TODOs	Summary/Remark
https://bugs.llvm.org/show_bug.cgi?id=45021				

Metrics

1. Number of landed patches
2. Successfully landed D41416
3. Improved startup time
4. Reduced memory consumption — ask Google to run the reimplemenation of D41416 on their builds
5. Stretch metric/goal — Build ROOT with -Druntime_cxxmodules=On on Windows

Optimizations

☐