Making Compilation Faster

Featuring Modules (Clang)

Mentor - Vassil Vassilev

Student - Shreyas Atre

Bit # 1 / 9

- Compilation of headers
  - Transitive (Successive)
  - N translation units, M headers ϵ each TU → M x N work
  - C++ compilation model involves templates → Huge code in headers

Bit # 2 / 9

- Modules (Clang header modules, Clang module map modules or Clang c++ modules)
  - Module compiled only once
  - Automatically translate #include directives into the corresponding module import
  - Module maps are the way to define links
  - Module.modulemap and no need to change headers
  - Are modules always faster? → No (Depends on “costs” of compilations)

Bit # 3 / 9

- Modules Overview
  - DAG of AST files
  - `clang -x c++-header -emit-module -o <directory>/<module name>.pcm -fmodules module.modulemap -fmodule-name=<module name> -std=c++17`
  - module "<module name>" {
  - export *
  - header "<header name>.hpp"
  - }

Bit # 4 / 9

- Modules Overview

  - DAG of AST files

  - `clang --std=c++17 main.cpp -fmodule-file={module name}=<directory>/{module file name}.pcm`

Bit # 5 / 9

- Patch Overview
  - Template Specializations
    - Load efficiently
      - Previous → Deserialized all the template specializations
      - Whenever specl. is requested→ check if it's already deserialized (LoadLazySpecializations)
      - Make this lookup faster by storing hash vs Decl. ID in a map (ASTReaderDecl)

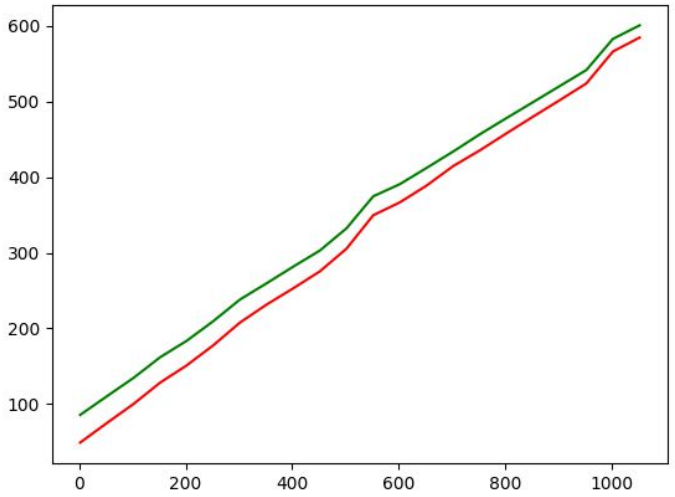
Bit # 6 / 9

- Patch Overview
  - Template Specializations
    - Store efficiently only for Modules
      - Instead of AddFirstDeclFromEachModule we AddFirstSpecializationDeclFromEachModule
      - Compute the ODRHash of template arguments and store it per specialization (ASTWriter)
      - Metadata - First Element in array → Number of Specializations, etc

Bit # 7 / 9

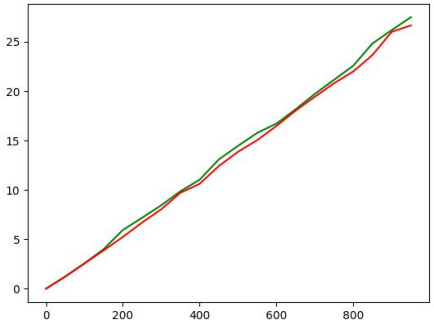
- Expectations

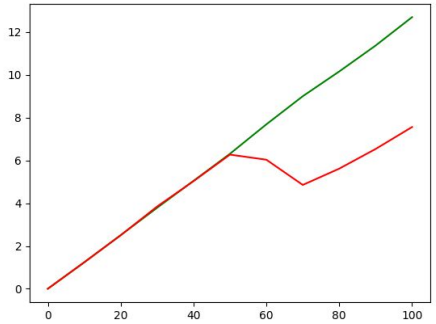
  - Reduced memory use
  - Memory usage in MB vs Number of Modules
  - red - patch (release), green - clang-14
  - Note - Graph is just for visualization and real nature maybe different



Bit # 8 / 9

- Expectations
  - Faster compilation times
  - Execution time (s) vs number of modules
  - red - patch (release), green - clang-14
  - Note - Graph is just for visualization and real nature maybe different





Bit # 9 / 9

❖ Summary

- ➢ Made lookup faster by storing hash vs Decl. ID in a map (ASTReaderDecl)
- ➢ A repository for experimentations → https://github.com/SAtacker/random-template-specls
- ➢ Yet to generate absolute and definitive results for this patch

❖ Reach -

- ➢ LinkedIn - https://www.linkedin.com/in/atreshreyas/
- ➢ GitHub - SAtacker
- ➢ satacker.github.io (not updated)
- ➢ Fall 2023 GRA → MS CS @ LSU, LA