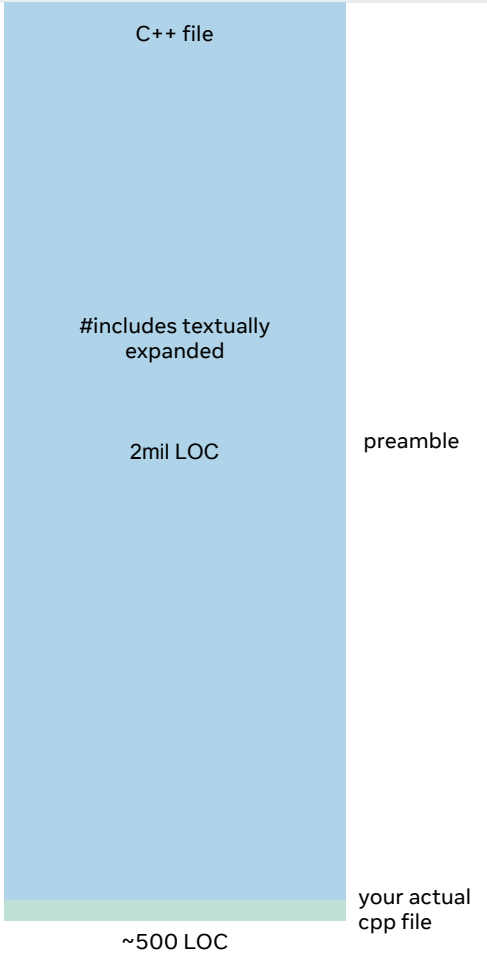
Improving clangd document open time with preamble caching

Alper Yoney
Dmitry Polukhin
Ivan Murashko
Kugan Vivekanandarajah

Challenges

- Huge monorepo with complex C++ files, many of them take more than a minute to compile
- Bad developer experience in IDE because clangd is not ready for a long time
- Headers combined can be millions lines of code but source itself is usually much smaller so clangd spends most of its time compiling headers
- Precompiled headers (PCH) could help but cannot in this case because clangd uses them internally to speed up re-compilation when the user modifies a file



Use Implicit C++ Modules as PCH Substitution

- **Solution:** generate C++ implicit modulemap and inject into compilation database (CDB) for clangd

- Dramatically speeds up file reopening time in clangd
- On the example file with 2MLOC: **120+ sec. -> ~3s**
- Overall median clangd document open time improvements: **more than 3X in our case**

Use Background Index for Prewarming Caches

- Naturally developers reopen only 20-30% of files in clangd so even if we speed up these files 20X, it won't give a lot
- Cache prewarming is required
- Clangd compiles all files from CDB once for building background index, usually it doesn't speedup file opening later because nothing is reused between these compilations
- With implicit modules in our setup, modules generated during background index are reusable on opening

Async Dynamic Indexing in Clangd

- Initial speedup of file reopening was “only” 2-3X
- Clangd builds dynamic index for the preamble and it causes visiting every node in AST and full deserialization that is slow
- Dynamic index can be built asynchronously in separate thread - see D148088 (implemented in collaboration with clangd community, special thanks to **Kadir Cetinkaya** and **Sam McCall**)
- Async dynamic index alone gives 20-30% document open time improvement even without implicit modules - already enabled in clang-17 by default and available to everybody
- With implicit modules speedup of document reopen is about 20X in P50

Next Steps

- Implement the preamble caching mechanism in clangd in upstream
- Produce preamble during background indexing
- Add options to control caching

Backup: Injected Compiler Options

- `-fmodules` - enable modules
- `-fno-implicit-module-maps` - don’t use implicit module maps because we resolve everything manually
- `-fmodule-map-file` - generated module map for all headers that exports everything
- `-fmodules-cache-path` - use separate path to the cache to control size and don’t interfere with build modules
- `-fmodules-validate-input-files-content` - check module invalidation by content hash instead of file timestamp
- `-Wno-module-import-in-extern-c` - help with extern “C” in headers
- `-Wno-error` - warnings in headers shouldn't break module compilation
- `-Xclang -fallow-pcm-with-compiler-errors` - produce serialized AST even in case of compilation errors in headers