“Does the win32 clang compiler executable really need to be over 21MB in size?”

Russell Gallop

April 2019

Back in the mists of time

- *"Does the win32 clang compiler executable really need to be over 21MB in size?"* is from an internal PlayStation®4 (PS4) bug filed in 2013

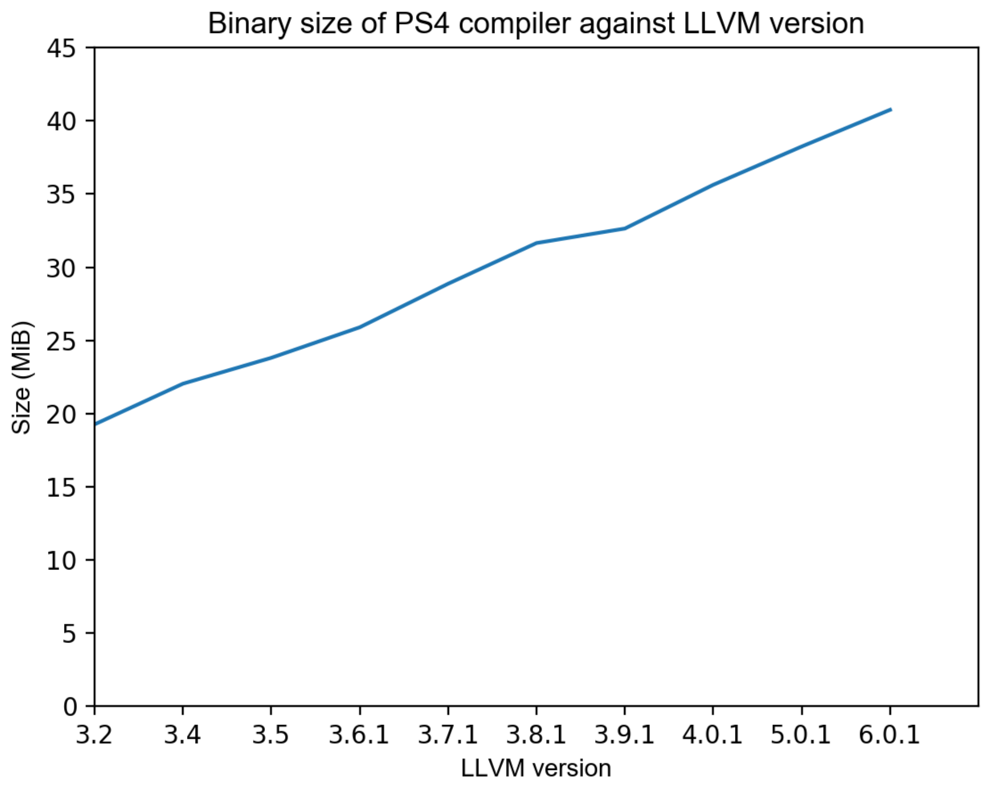
- The original PS4 compiler was about 3 times larger than the proprietary PlayStation®3 compiler

- That was based on LLVM 3.2

Today

- The PS4 compiler based on LLVM 6.0 is about 40MiB

- This includes many new features PS4 developers appreciate:
  - LTO
  - PGO
  - Diagnostics
  - C++14/17
  - And more...



But what do we “really need”?

- The PS4 compiler needs to support:
  - Two languages: C/C++
  - One target triple: `x86_64-scei-ps4`, one cpu: `btver2`
  - One object format: ELF

- http://llvm.org assures us that:
  - *“The LLVM Project is a collection of modular and reusable compiler and toolchain technologies.”*

- Building just the features we need:
  - Keeps build times down
  - Simplifies testing

- So how close can we get to just doing that?

Method

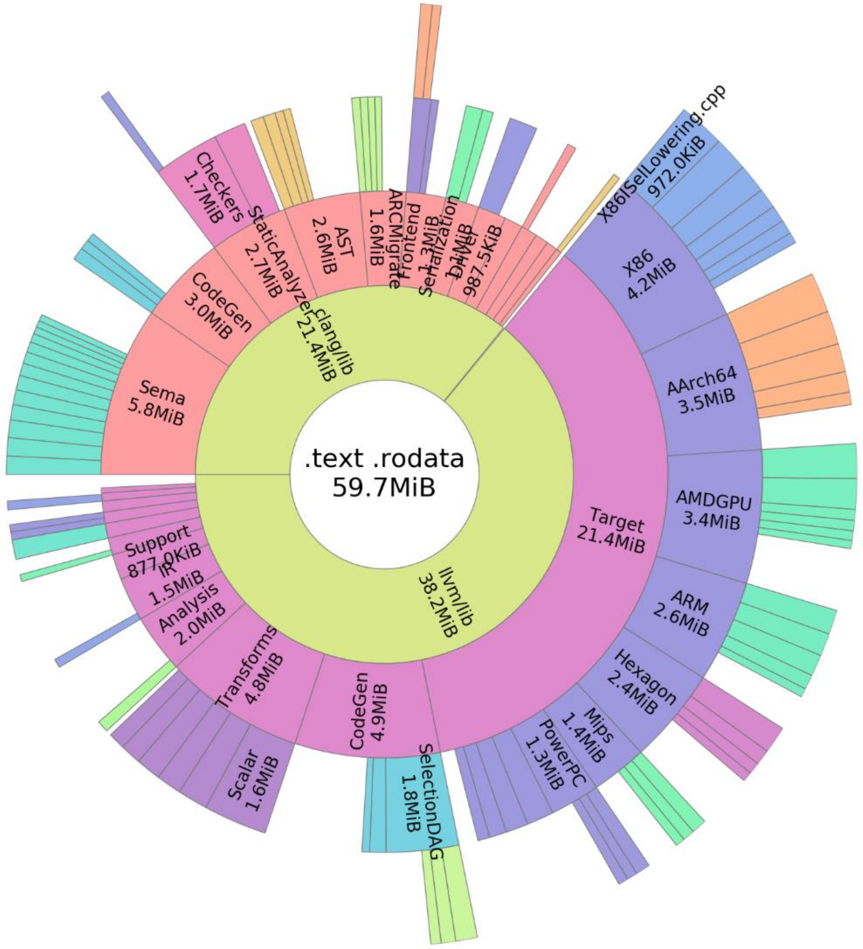
- Analyzed binary size of opensource LLVM using Bloaty McBloatFace* on Linux
  - RelWithDbgInfo build config
  - Just `.text` and `.rodata` sections as they are largest in Release configuration

- Not exactly the same as Windows binary size but similar ballpark

* I consider “bloat” as anything our customers don’t need in the executable. No offence intended!

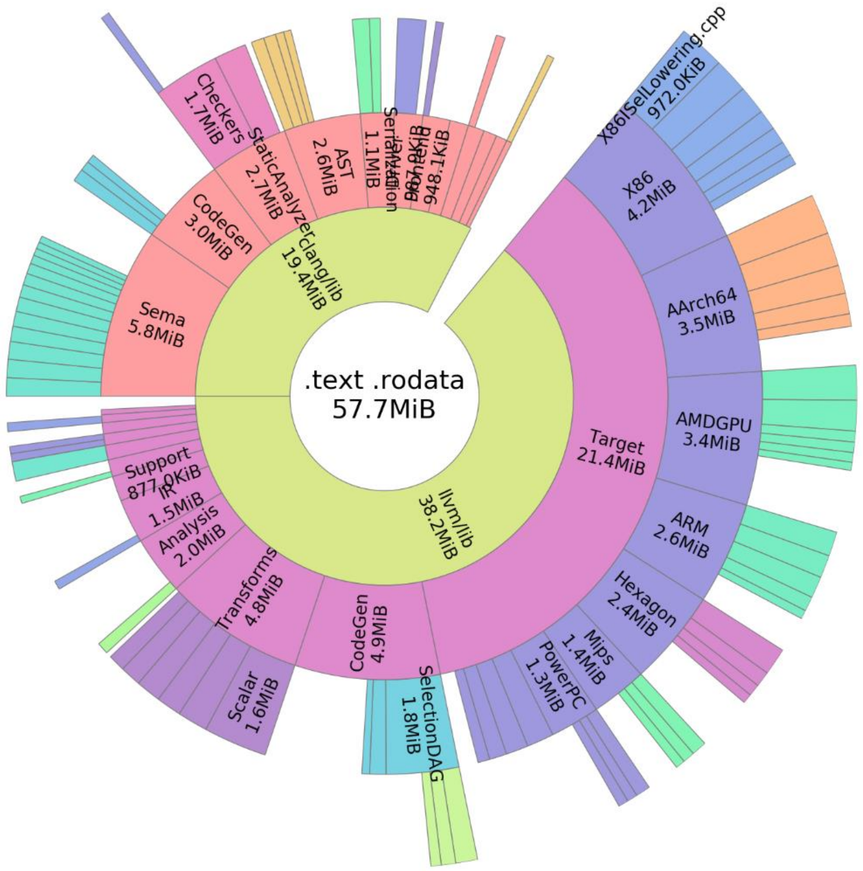
Configuration	
Build configuration	RelWithDbgInfo
OS	Ubuntu 18.04
Host toolchain	clang 6.0
llvm-project.git revision	llvmorg-8.0.0-rc5

Full build



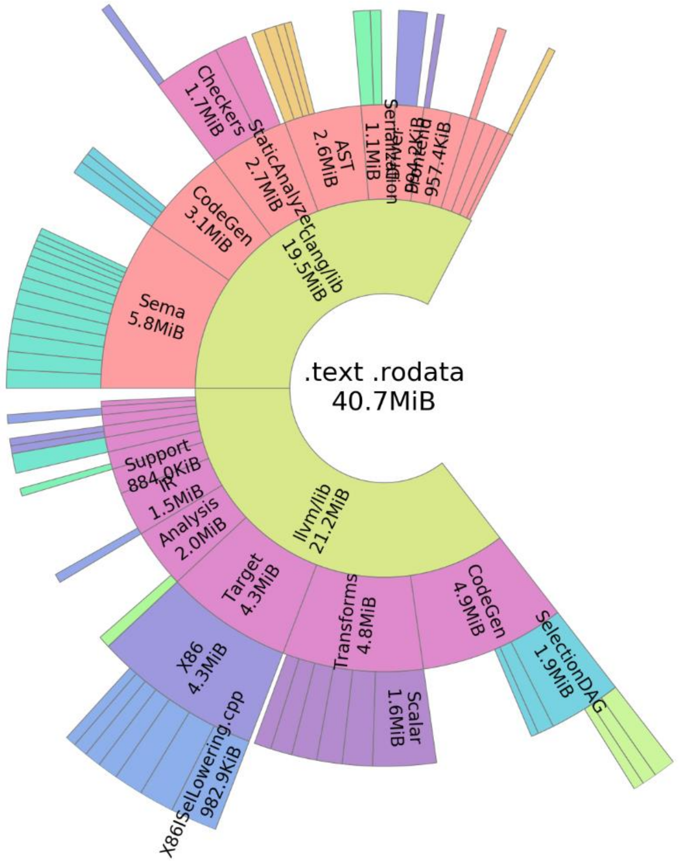
- This is a breakdown of a full build of bin/clang by folder and compile unit

Just C/C++



- Only CMake option to disable unused language features is:
  - `-DCLANG_ENABLE_ARCMT=OFF`
- This saves about 2MiB
- Based on strings in filenames this leaves:
  - *"ObjC" – 1.05MiB*
  - *"OpenMP "– 1,002KiB*
- Both of these are hard to remove
  - Can't be removed just by changing CMake options or files

Just C/C++, Just X86

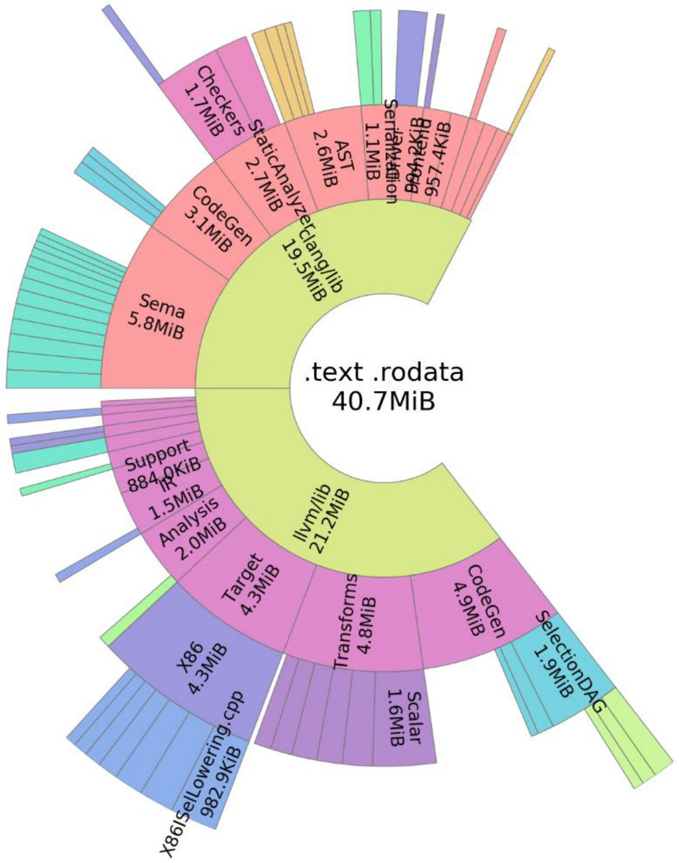


- We can disable backends other than X86, saving ~17MiB, about 30%

- Built with:
  - `-DCLANG_ENABLE_ARCMT=OFF`
  - **`-DLLVM_TARGETS_TO_BUILD=X86`**

- This still leaves
  - *Other toolchains ARM, PPC etc. (clang/lib/Driver/ToolChains) – 350KiB*
  - *Other targets (clang/lib/Basic/Targets) – 177KiB*
  - *Global ISel – 195KiB*
  - *> 100 subtargets - ??*

Just C/C++, Just X86, Just ELF



- Don’t believe that we can easily disable other object file formats so no change

- Built with:
  - `-DCLANG_ENABLE_ARCMT=OFF`
  - `-DLLVM_TARGETS_TO_BUILD=X86`

- So we still support
  - *Other object formats (MachO, Wasm, COFF etc.) - 193KiB*
  - *Codeview debug - 160KiB*

Summary

- *“Does the win32 clang compiler executable really need to be over ~~21~~ 40MB in size?”*
  - Probably not!
- LLVM is modular in many ways but not in all ways that you might want
  - Scaling down to a subset of features is not always easy
- LLVM just keeps growing ☺
  - As LLVM grows modularity becomes even more important
- We should continue to look for ways to make LLVM more modular