Valgrind & Debuginfo

Mark J. Wielaard

Fosdem 2022

Why debuginfo

- Translating binary to source and back
- For user to provide symbolic information
  - Suppression files
- For valgrind to addresses as symbol names and source lines
  - Annotate backtrace with symbols and source lines
  - Inline information (--read-inline-info=yes)
  - Variable locations (--read-var-info=no)

(*) For now we ignore unwind tables,
demangling and non-DWARF (PDB)

What debuginfo

- Symbol tables
  - .dynsym, .symtab (in.debug)
  - Address (range) to symbol name
- Line table
  - .debug_line
  - Address to source file & line
- CUs (Compile Unit) and DIEs (Debug Info Entry)
  - .debug_info/.debug_abbrev (tree of DIEs)
  - Program scope (inlines, .debug_ranges)
  - Variables & locations (.debug_loclist)
  - Types

Where debuginfo

- Binary itself
- Separate .debug file
  - Found through /usr/lib/debug/.build-id/
  - Found through .gnu_debuglink section
- valgrind-di-server (valgrind --debuginfo-server)
- debuginfod, if DEBUGINFOD_URLS env set

Debuginfod support

- Since valgrind 3.18.1, patch by Aaron Merey
- Spawns debuginfod-find for build-id (*)
  - Creates a cache to get .debug file quickly
    $XDG_CACHE_HOME/debuginfod_client/
- Some distros now set DEBUGINFOD_URLS by default
- Federating server https://debuginfod.elfutils.org/

(*) https://bugs.kde.org/show_bug.cgi?id=445011

valgrind DWARF reader (was) slow

- Exposed by debuginfod support

  Suddenly there always was debuginfo for everything
- C++ hello world (linked against libstdc++)

  Default valgrind (memcheck) with debuginfo (100MB total!)
  - Before ~12 seconds
  - After ~0.45 seconds
  - Without debuginfo ~0.25
  - Without valgrind ~0.005

Why is/was reading debuginfo slow?

https://bugs.kde.org/show_bug.cgi?id=442061

- Fully skip CUs and children of DIEs without addresses
- Don’t read line tables for CUs without addresses
- Reuse of line tables and abbrevs (dwz)
- Lazy reading of abbrevs

What more can be done?

- There are still two DWARF readers
- DWARF6 might introduce multi-level line-table
- Even more lazy reading (read CU on first use of address)
  - Use .debug_aranges
- Only do --read-var-info reading on error reporting
  - But DWARF info is wrong way around
- Make debuginfod-find only read “chunks”

  like valgrind-di-server (or get rid of chunks?)
- Long running debuginfod-find (keep connection)