Collabora Productivity

Reducing Build Time

By Luboš Luňák

Software Developer at Collabora Productivity

LibreOffice is a large C++ codebase

Long build times

- Clean builds, when a library changes, ...

Long developer cycle

- Modify-build-run/test/debug

Large disk usage

Get a better computer

Faster CPU

More RAM

Faster HDD (=SSD)

Get more build power

Distributed build (C++ compiles)

- Use a second (or more) computers
- Icecream
- --enable-icecream
- --with-parallelism=<num>

Cache builds

Cache repeated C++ compiles

- Ccache
- --enable-ccache (default)
- Helps only when repeating exactly the same compile
- Switching branches, git bisect, git revert, make clean
- ~10% overhead
- Set sufficient cache size
- Export CCACHE_COMPRESS=1 to reduce disk usage

Compiler

GCC / Clang

- Library_sm (starmath/)
- GCC7: 28s
- Clang6: 26s
- YMMV

Clang compiler plugins

- Clang6 –enable-compiler-plugins: 39s
- make COMPILER_PLUGINS=
- Upstream checks, reduce number of default checks?

Linker

BFD ld / Gold / LLD

- libsclo.so (--enable-dbgutil)
- --enable-ld=<ld>
- BFD: 32s
- Gold: 10s
- LLD: 9s (works only with Clang)

Split debug

Split DWARF

- --enable-split-debug
- libsclo.so (--enable-dbgutil)
- Normal: 450M, 10s (gold), 32s (BFD)
- Split: 203M, 6s (gold), 20s (BFD)
- Full dbgutil build: 20G vs 15G

Gdb symbol loading

Gdb startup time

- Calc gdb attach time : 51s
- --enable-gdb-index
- Calc gdb attach time: 16s
- libsclo.so 136M (split+index vs 203M split debug)

Precompiled headers (PCH)

- --enable-pch
- Tricky to use (what to put in the PCH)
- Additional disk space needed
- MSVC – helps a lot
- Clang – helps, less
- GCC - ??
- YMMV

Build only what needs building

Build only a module/library

- make sc
- make Library_sc
- make -C sc (-j<cpu> -s)

Do not need to always run tests

- make → make build-nocheck
- make sc → make sc.build

Do not rebuild what will not change

If e.g. a header included by many source files changes

- touch -t0101010101 header.hxx
- make will not rebuild files depending on an old header
- Safe only if no code changes or is binary compatible
  - Non-virtual function added, modified, …
  - e.g. ‘KDE binary compatibility’ for a full list

Unit Tests

Sometimes using unit tests saves time

- Modify → build → run LO → test → modify
- Running LO and testing manually takes time
- Create a unit test for the tested/developed area
- Modify → build&run test → modify
- make CppunitTest_sc_ucalc CPPUNIT_TEST_NAME="xx”

Various

- No compiler optimizations (--enable-debug at least)
  - Developers should use –enable-dbgutil anyway
- --without-doxygen
- --without-java
- --with-system-libs --with-system-headers

Improve code

- #include only what is needed
- Forward declarations instead of #include
- Avoiding templates in headers
- …
- But this is a lot of work.
-

Edits (added after talk)

Some features work better with new tools

- GDB index, split DWARF, etc. work the best (or at all) with the latest versions of the relevant tools (such as GDB and linker)

GDB load times

- Auto-loading of symbols can be disabled
- set auto-solib-add off (add to ~/.gdbinit)
- Symbols can be loaded manually (and selectively) as needed
- share (sharedlibrary, see ‘help share’ in gdb)

Collabora Productivity

Thank you.

By Luboš Luňák

l.lunak@collabora.com