llvm-buildmark

Observations, tips, and tricks on reducing LLVM build times

Alex Bradbury asb@igalia.com

EuroLLVM 2023, 2023-05-11

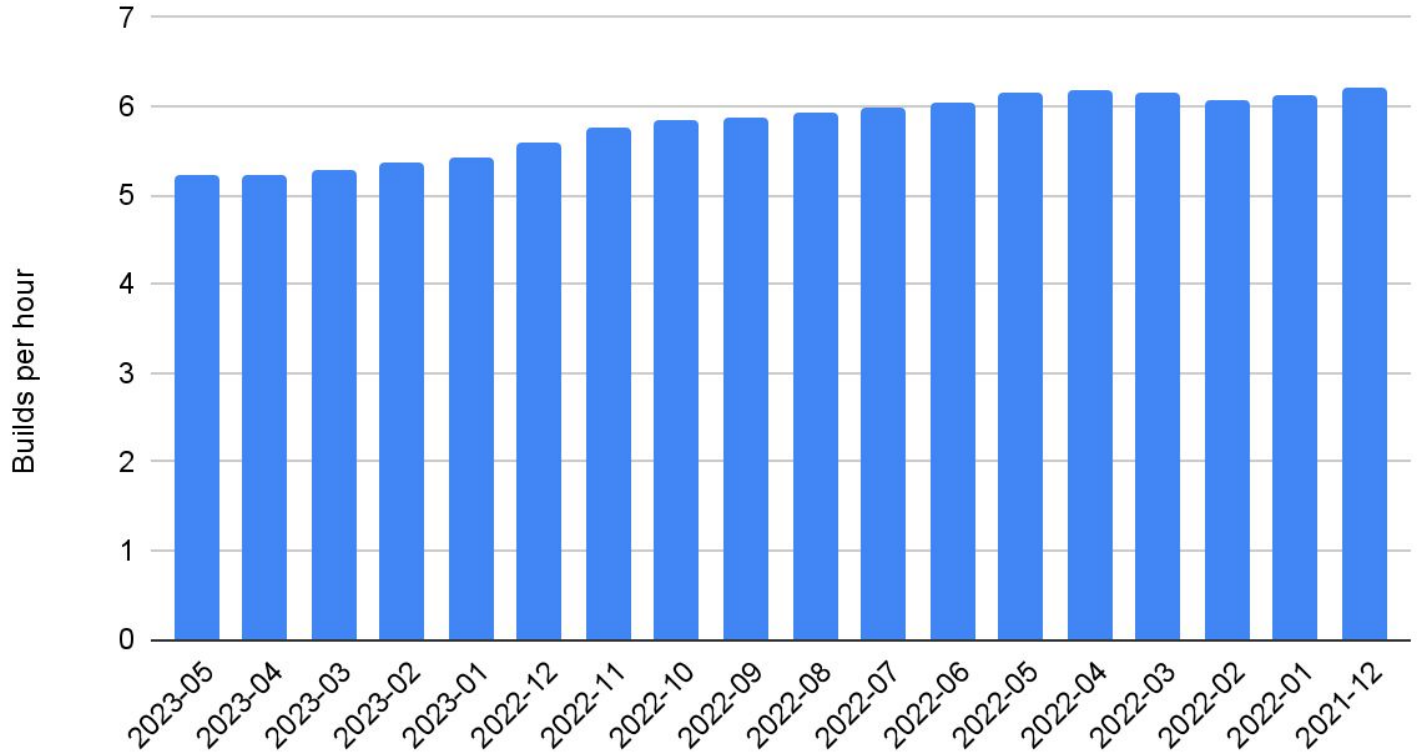
igalia

Progression of LLVM build times

```
mkdir build && cd build
cmake -G Ninja -DCMAKE_BUILD_TYPE="Release" \
  -DLLVM_ENABLE_PROJECTS="clang;lld" \
  -DLLVM_ENABLE_ASSERTIONS=OFF \
  -DLLVM_CCACHE_BUILD=OFF \
  -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ \
  -DLLVM_ENABLE_LLD=True \
  -DLLVM_TARGETS_TO_BUILD="all" \
  ../llvm
```

- Ryzen 9 5950x host (16c/32t, Zen3, 3.4GHz base clock, 4.9GHz boost clock)
- Arch Linux, recent standard packages unless otherwise specified
- 128GiB RAM, NVMe SSD
- Warning: all figures are rough and “for fun” - measurement errors very possible.

Builds per hour over time



16.0.3 “Release” build comparisons

- Clang (15.0.7)
  - 5.38 builds/h
- GCC (12.2.1)
  - 4.29 builds/h
  - Roughly the same for -fuse-ld={bfd,gold,lld}

“Debug” build comparisons

```
cmake -G Ninja -DCMAKE_BUILD_TYPE="Debug" \
  -DLLVM_ENABLE_PROJECTS="clang;lld" \
  -DBUILD_SHARED_LIBS=False -DLLVM_USE_SPLIT_DWARF=False \
  -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ \
  -DLLVM_ENABLE_LLD=True \
  -DLLVM_TARGETS_TO_BUILD="all" \
  ../../llvm
```

16.0.3 “Debug” build comparisons

- Default
  - 5.18 builds/h (4.66 builds/h with ld.bfd)
- With shared libs
  - 5.35 builds/h
- With split dwarf
  - 5.21 builds/h
- With split dwarf and shared libs
  - 5.36 builds/h
- Release (copied from previous slide)
  - 5.38 builds/h

Debug incremental build scenario 1 (single backend's C++ file)

- Default
  - 100 ibuilds/h (14.1 ibuilds/h with ld.bfd)
- With shared libs
  - 424 ibuilds/h
- With split dwarf
  - 124 ibuilds/h
- With split dwarf and shared libs
  - 450 ibuilds/h

Debug incremental build scenario 2 (Intrinsics.td change)

- Default
  - 7.63 ibuilds/h
- With split dwarf and shared libs
  - 8.16 ibuilds/h

Further optimisations (not explored here)

- Build less
  - Disable targets or subprojects you don't care about. See -DLLVM_TARGETS_TO_BUILD, -DLLVM_ENABLE_PROJECTS
- Ccache
  - -DLLVM_CCACHE_BUILD=ON or
    -DCMAKE_C_COMPILER_LAUNCHER=ccache
    -DCMAKE_CXX_COMPILER_LAUNCHER=ccache
- Non-debug TableGen
  - -DLLVM_OPTIMIZED_TABLEGEN
- Reduce relinking
  - LLVM_APPEND_VC_REV
- Faster host compiler
  - PGO, PGO+Bolt, -march=native
- Other
  - See https://llvm.org/docs/CMake.html

Future

Watch muxup.com/llvm-buildmark for scripts, further experiments, benchmarks on different hardware.