BOLT'ED CLANG: HOW GOOD IS IT ON AARCH64?

ELVINA YAKUBOVA, SJOERD MEIJER

EUROLLVM 2025

HOW GOOD IS BOLT ON AARCH64?

Questions we wanted to investigate:

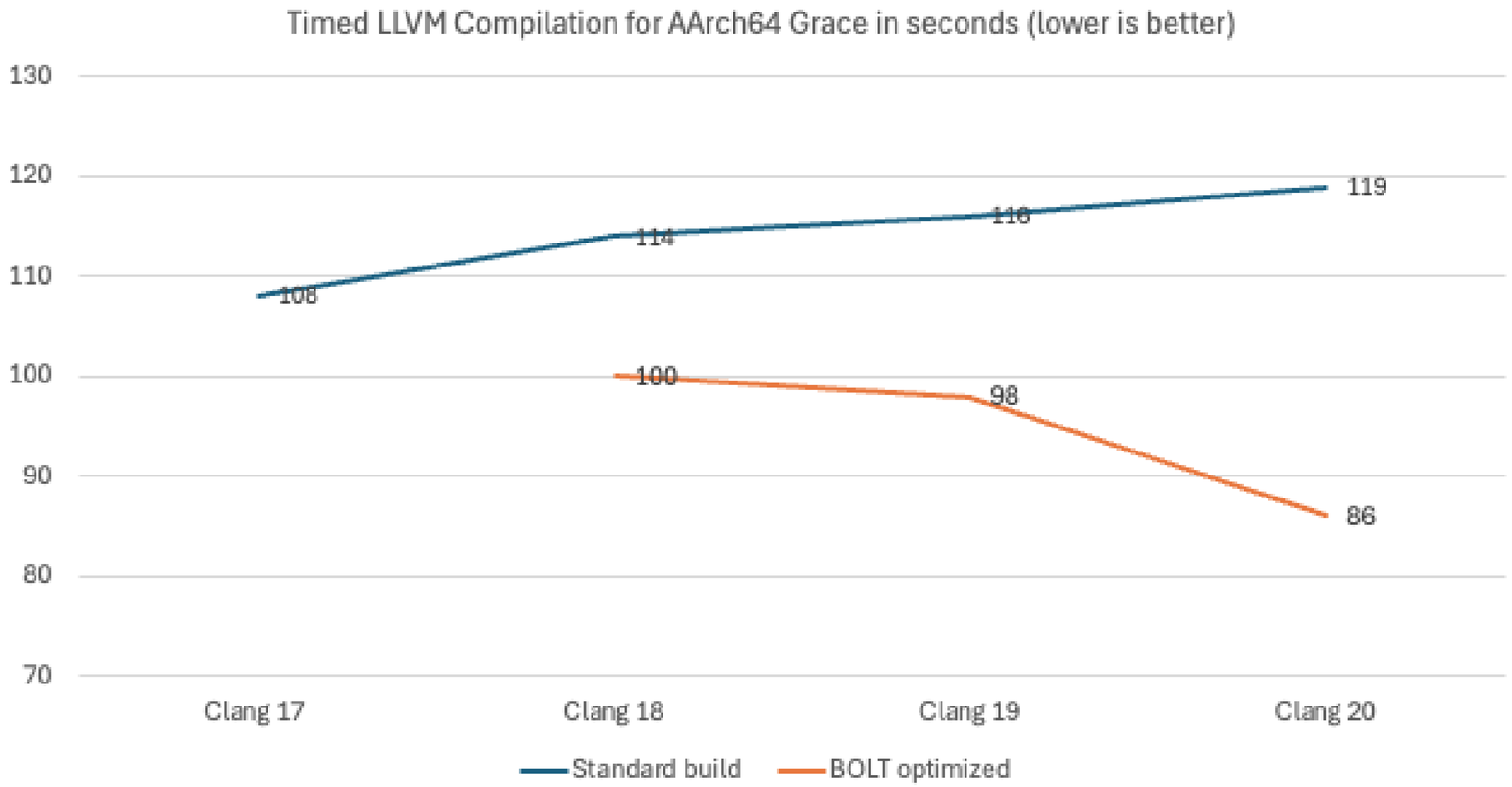
1. What performance improvements can we obtain by using BOLT on AArch64 platforms?
2. Should we BOLT our Clang build?
3. And can we do better and change build process?

We will show:

- Performance results for timed compilation of LLVM and CTMark (LLVM test-suite)
- The trends for Clang-18, Clang-19, and Clang-20
- Show experiments with more and different profiles

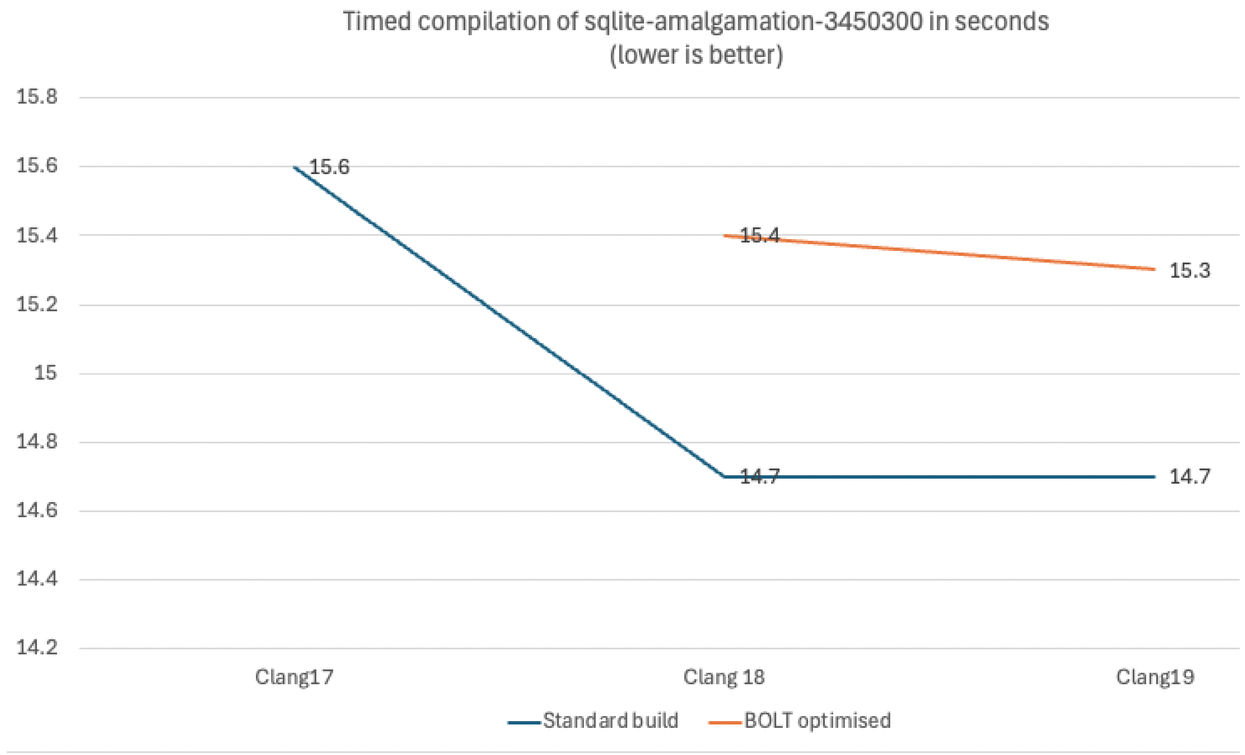
BOLT'ED CLANG: LTO/PGO/BOLT OPTIMISED COMPILER

- BOLT optimised AArc64 Clang toolchain
  - A.k.a.: how do we create a fast compiler?
  - Metric: measure compilation time of the BOLT'ed compiler: 27% speed-up of BOLT'ed Clang-20 compared to a standard build:



IS IT GOOD FOR ALL WORKLOADS?

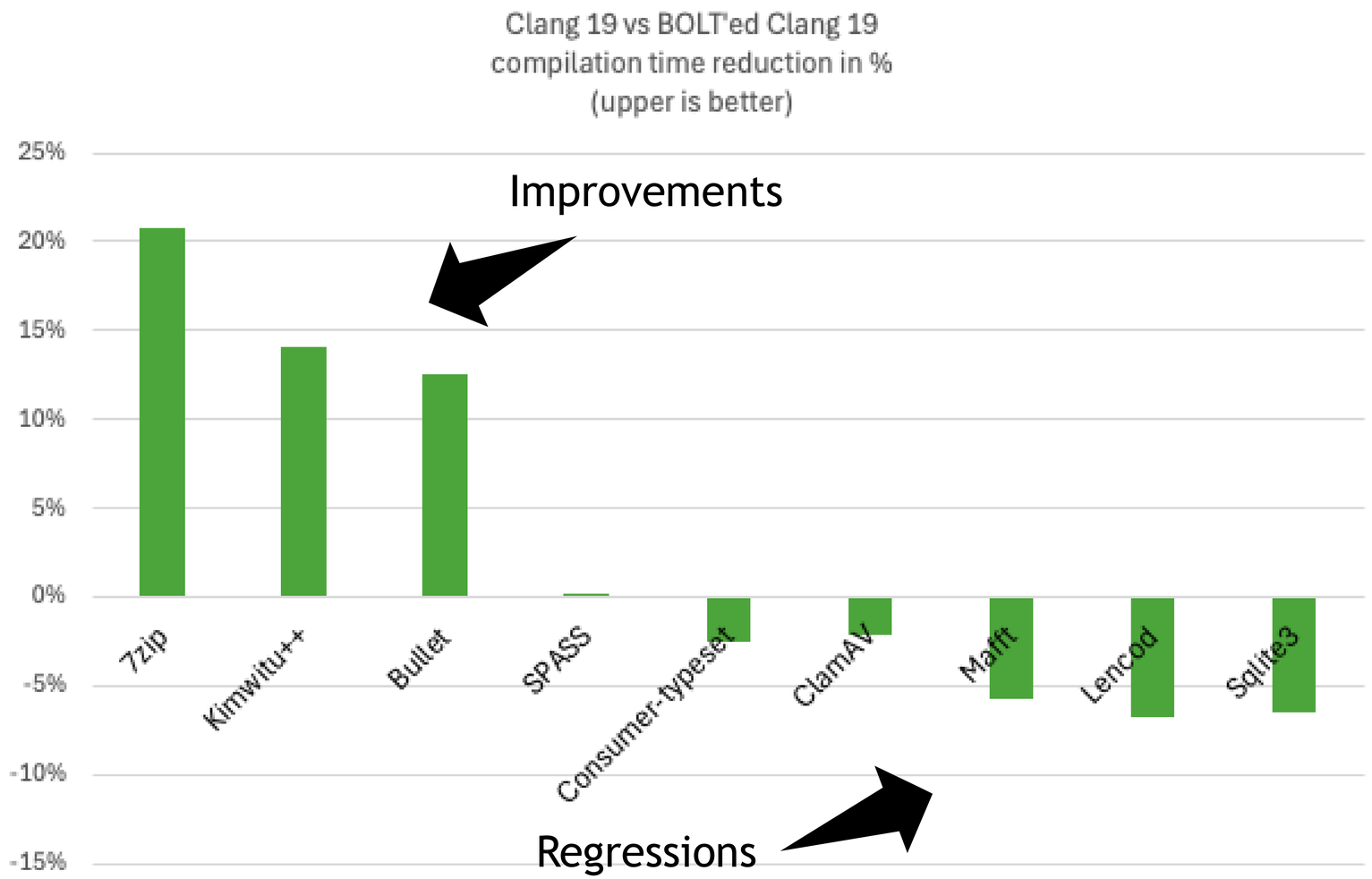
- 4% compile-time regression for SQLite with BOLT'ed Clang:



Hypothesis:

- BOLT'ed Clang is trained on a modern C++ code (LLVM),
- Maybe this works less well for SQLite that is C-code.

IT'S NOT ALL GOOD, MORE REGRESSIONS...

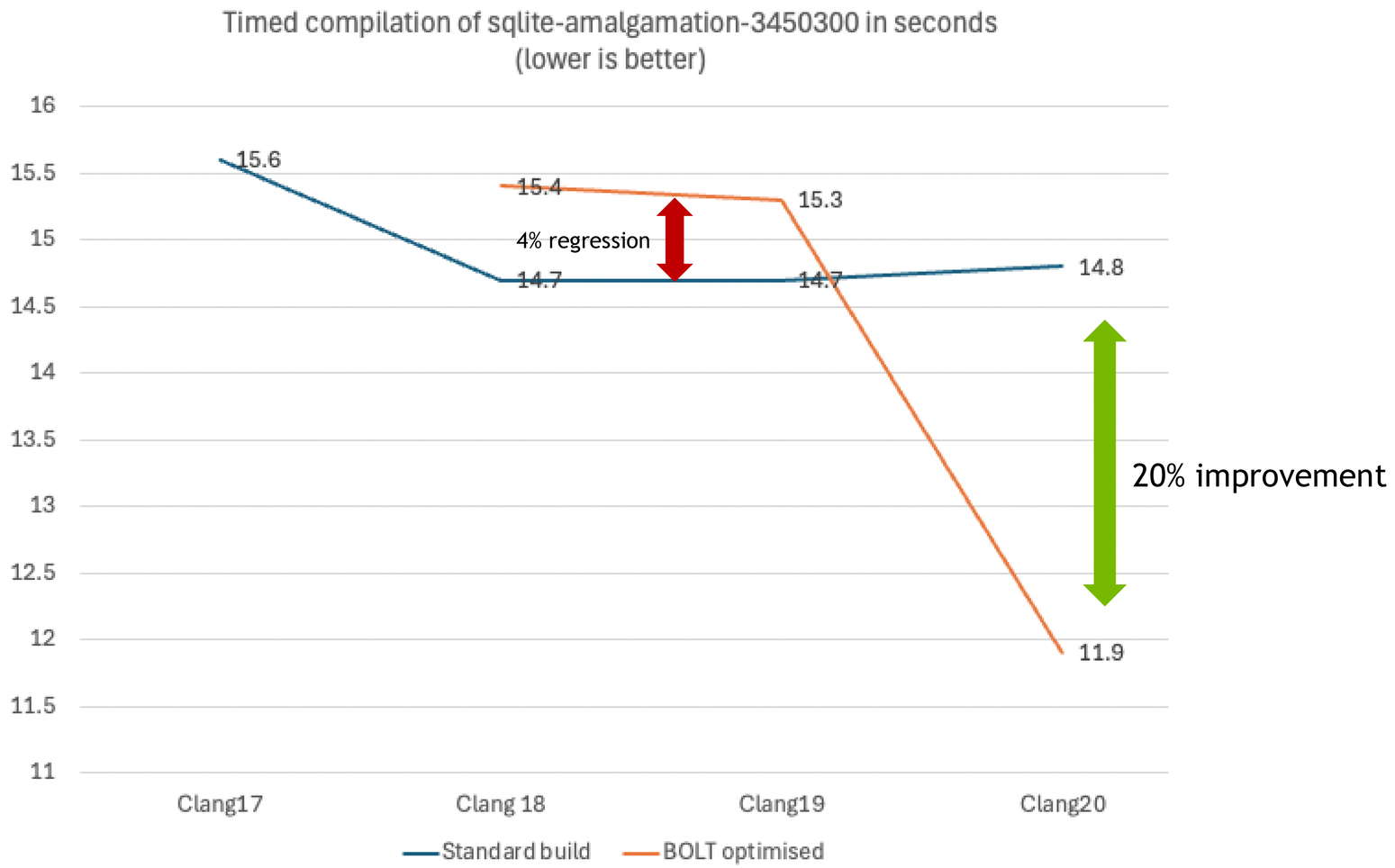


CTMark C-code base:

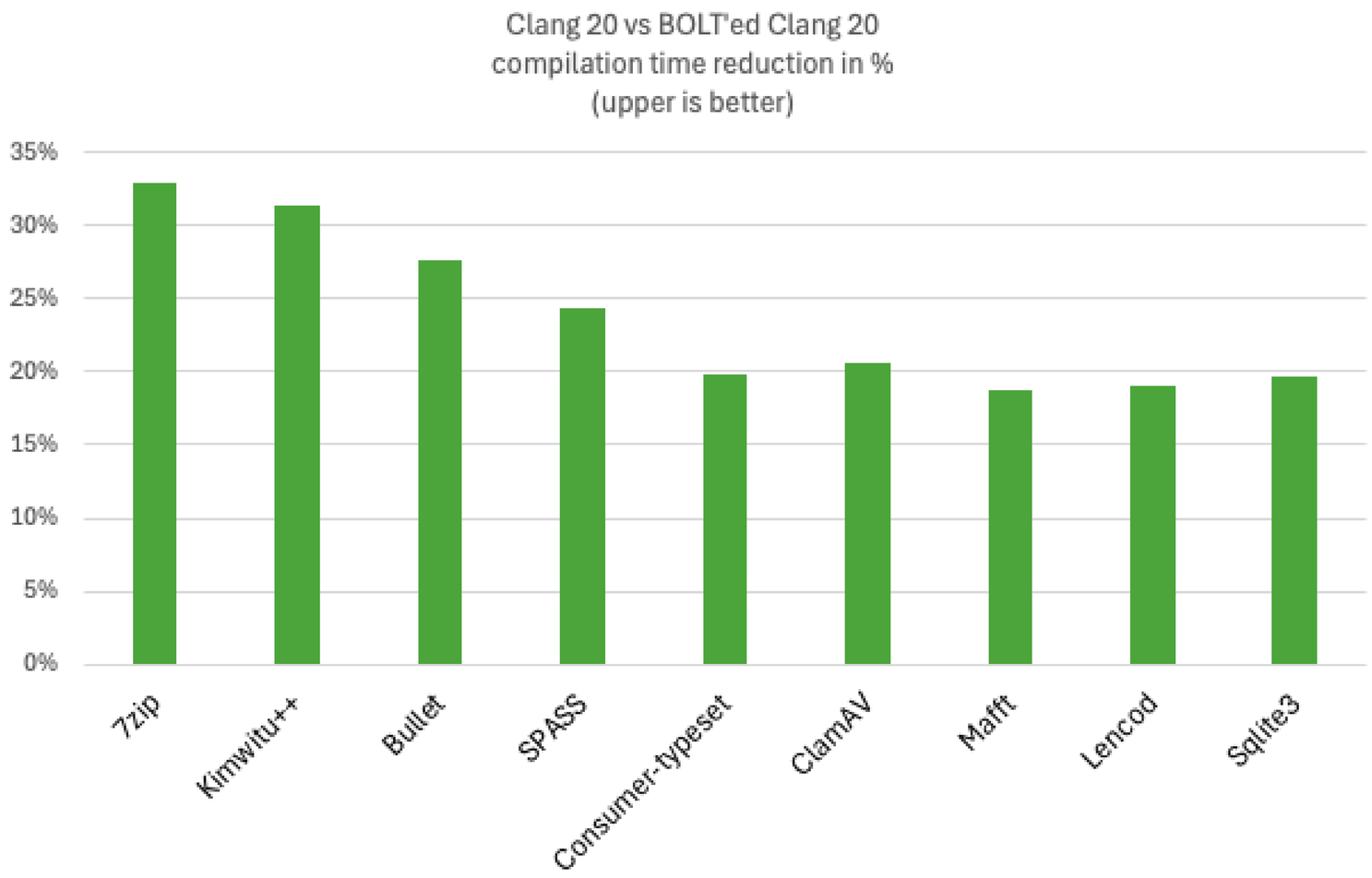
- SPASS
- Consumer-typeset
- ClamAV
- Mafft
- Lencod
- Sqlite

CLANG 20: GAME CHANGER

- Massive 20% SQLite compile-time improvement!



ONLY IMPROVEMENTS WITH CLANG 20



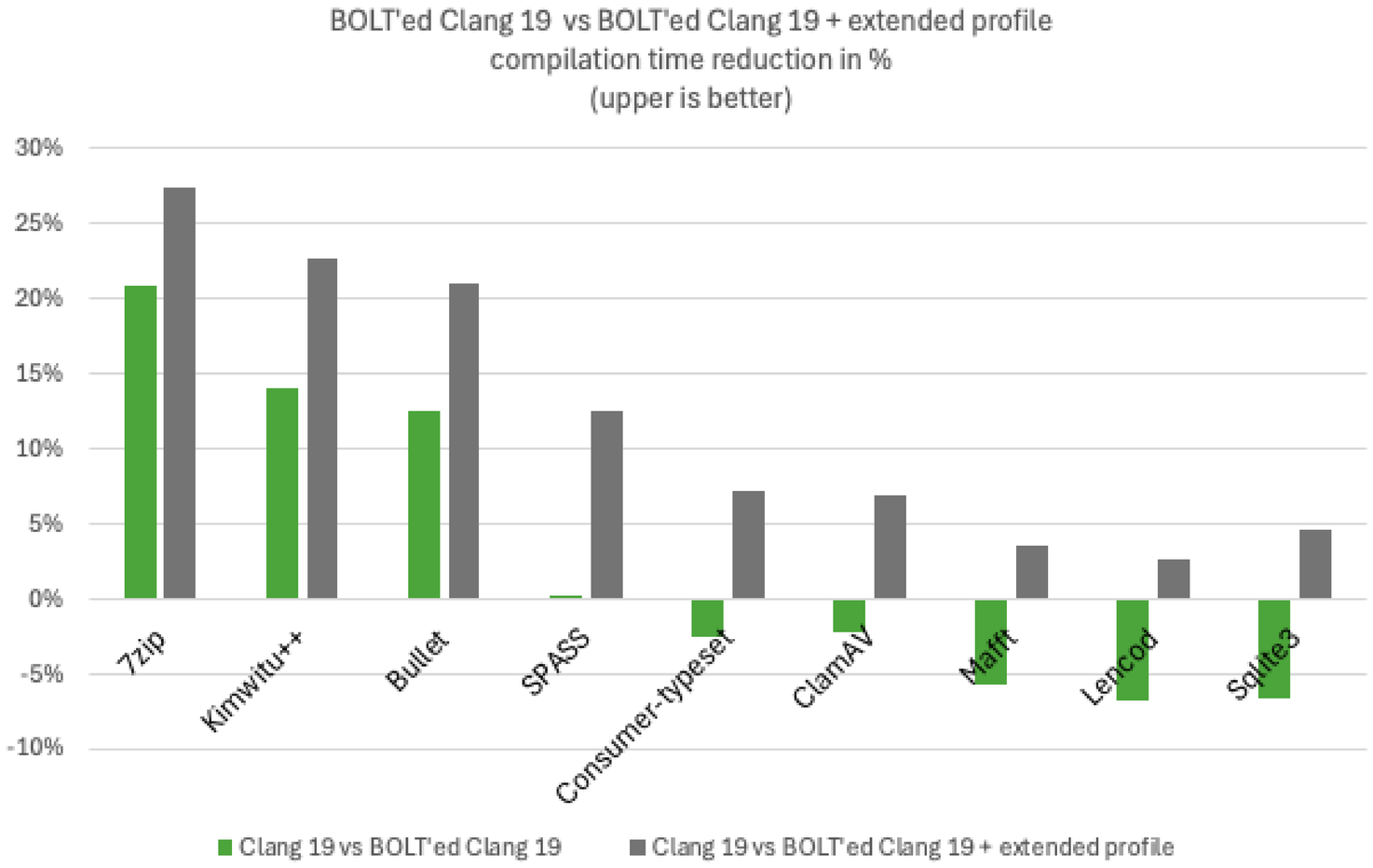
CAN WE DO BETTER?

- BOLT'ed Clang-20 improvements could come from:
  - CMake configurations: different options passed on,
  - BOLT learned new optimisations,
  - More/better profiles?

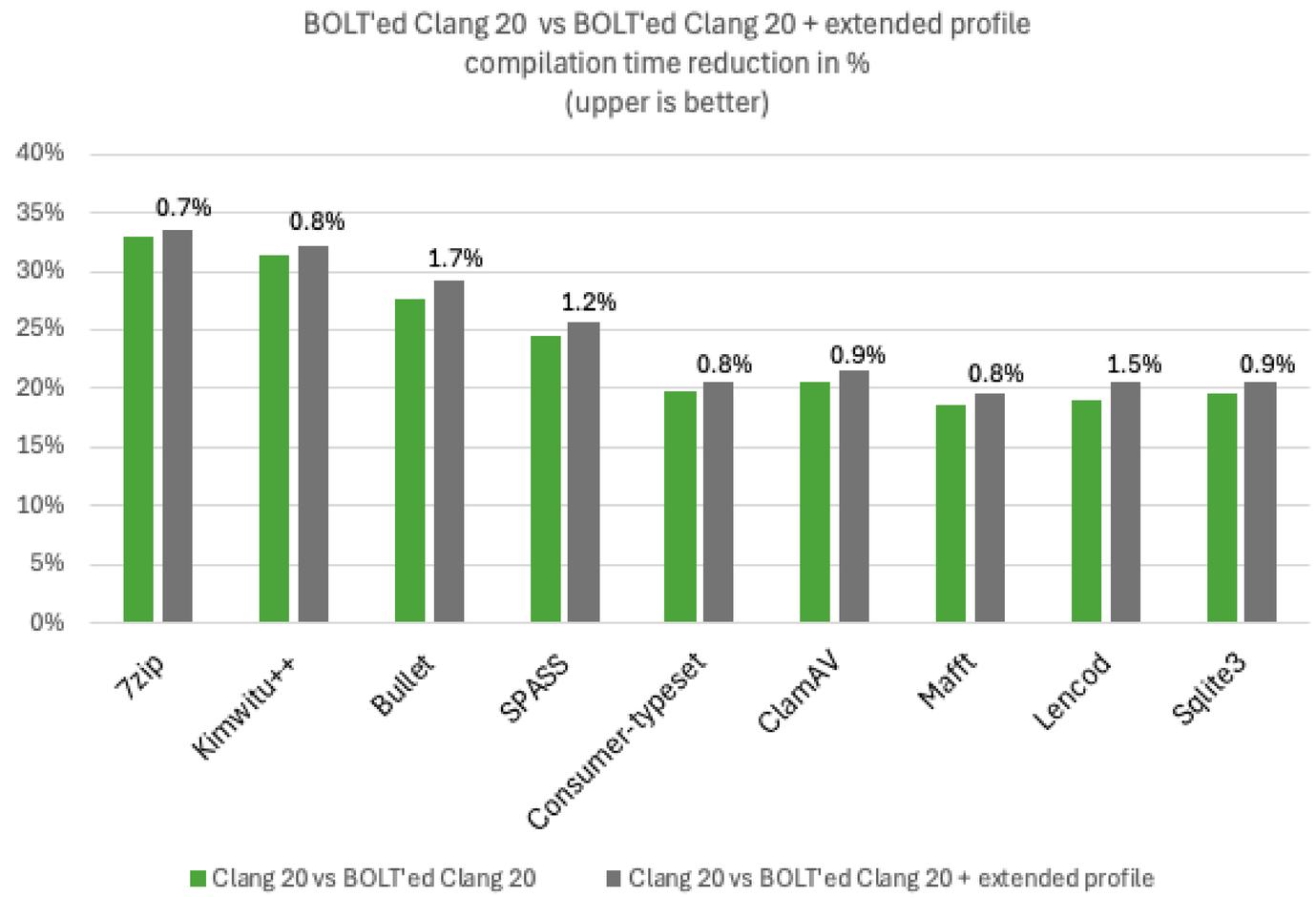
- Can we do better?
  - BOLT'ed Clang is trained on a modern C++ code-base (LLVM),
  - Should we extend the training stage with more/different profiles?

- Extended Profile:
  - Collect profiles for CTMark from the LLVM test-suite (C code-base),
  - Collect profiles for LLVM (C++ code-base),
  - Merge LLVM + CTMark profiles and use that as input to BOLT.

MORE PROFILES FIX THE CLANG 19 RESULTS

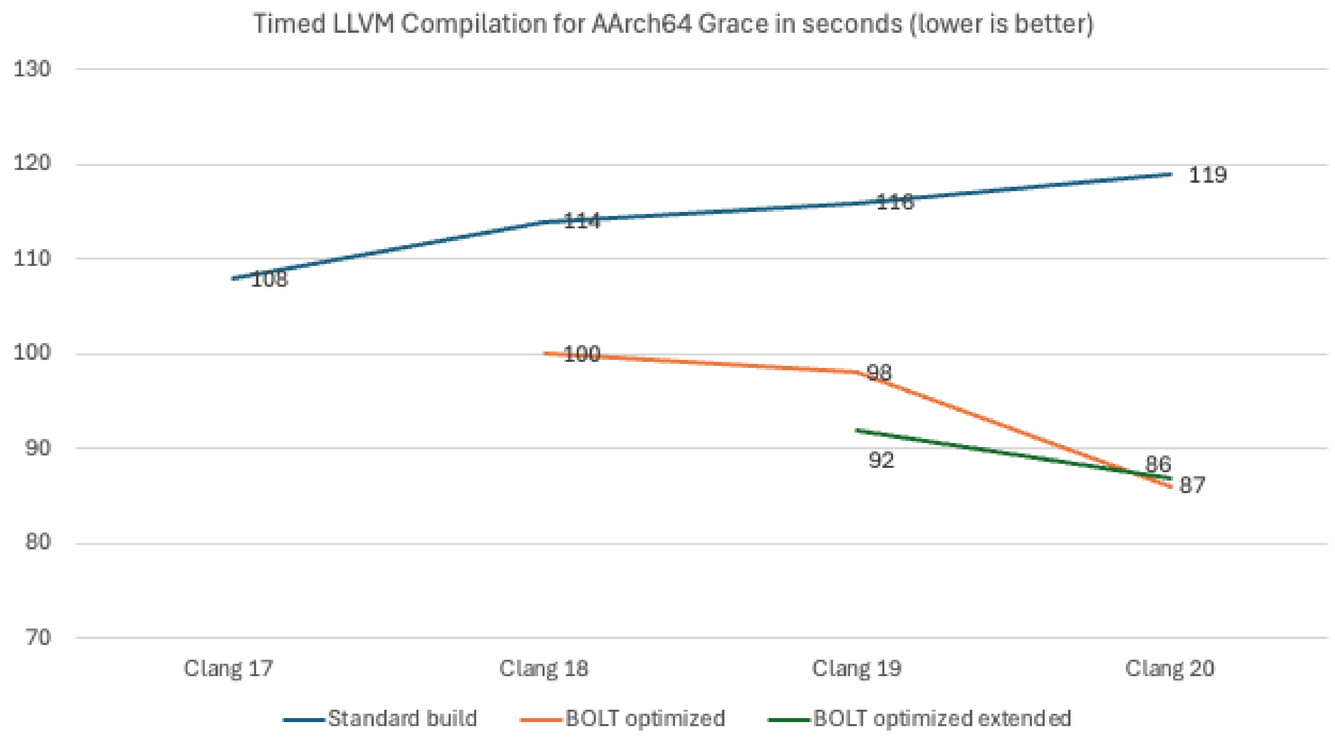


EXTENDED PROFILES: ADDITIONAL IMPROVEMENTS LESS THAN 2%



LTO/PGO/BOLT OPTIMISED CLANG COMPILER

- Additional 6% improvement on Clang 19
- No difference on Clang 20



CONCLUSIONS

- Clang-20 and BOLT-20 are great, also on AArch64!
  - BOLT'ed Clang-20 is 27% faster than Clang-20
  - Universally good: the same or better performance (for LLVM and CTMark)
  - Fixes the issues with Clang-19 and older versions that were not so great yet.

- Yes, Clang releases should be BOLT'ed
  - They started with the latest release

- Extended profiles:
  - Clang-20 is now also trained on libLLVMSupport: big improvement
  - Training it even more with CTMark: almost makes no difference!
  - The current CMake Clang/BOLT build process and configuration is enough

- Future work:
  - CTMark apps are small, investigate more/bigger apps,
  - With extended profiles, they should also be added to the PGO stage (i.e. not only BOLT)


Thank you
NVIDIA