Comparative Analysis of Linking Efficiency

Why is LLD not as fast as mold?

Anna Szymkowiak - A.M.Szymkowiak@student.tudelft.nl
Responsible Professor: Soham Chakraborty
Supervisor: Dennis Sprokholt
TU Delft

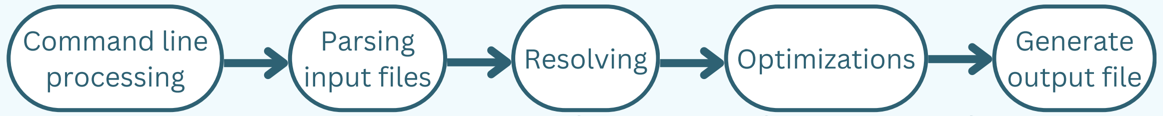
1. BACKGROUND

The Compiler Toolchain [1]:

- compiling source files into object files;
- linking those object files into a single executable.

Linking - time-consuming when managing a large number of object files. Improving the efficiency of this phase → reduce the overall build time for large projects (important during software development).

Linking process [2]:



Command line processing → Parsing input files → Resolving → Optimizations → Generate output file

Linker script controls the allocation of sections from input files in the output file [3].

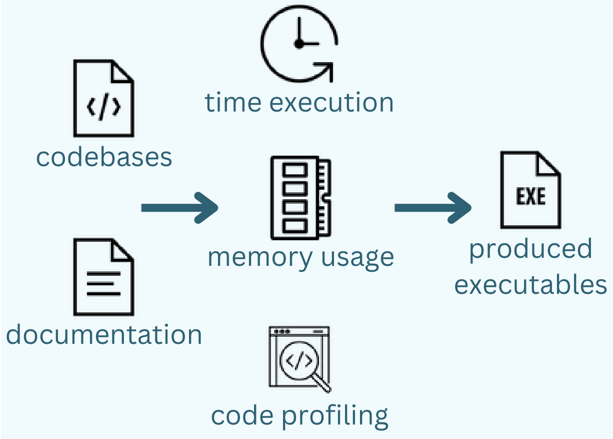
This research aims to introduce linking into academic discussions by analyzing and comparing two linkers - **LLD** and **mold**.

2. RESEARCH QUESTIONS

- What does the linking process look like in LLD and in mold and what are the differences?
- What are the differences in architecture between LLD and mold?
- What factors contribute to mold's performance?

3. METHOD

Compare **LLD** and **mold**:



codebases

documentation

→ time execution

memory usage

code profiling

→ produced executables

6. FUTURE WORK

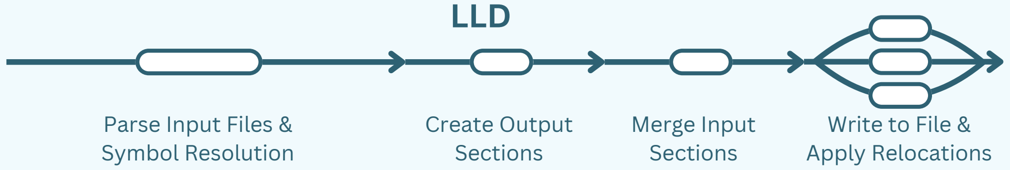
- More thorough comparison of produced executables.
- Compare performance of mold on the projects that output multiple executables.
- Research into constraints imposed by linker scripts.

7. LIMITATIONS

- Complex codebases → requires to make assumptions.
- Linking - seemingly a straightforward process, yet complicated - a lot of details that need to be taken into account.

4. RESULTS

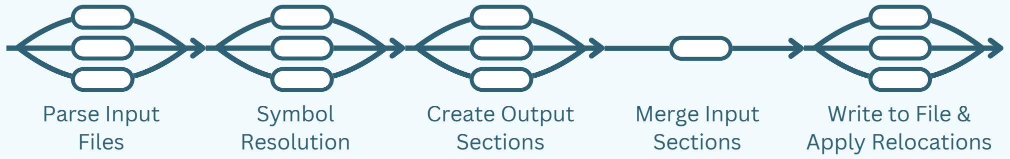
LLD



Parse Input Files & Symbol Resolution → Create Output Sections → Merge Input Sections → Write to File & Apply Relocations

Figure 1: Simplified overview of the linking process in LLD, highlighting parallel or sequential processing

mold



Parse Input Files → Symbol Resolution → Create Output Sections → Merge Input Sections → Write to File & Apply Relocations

Figure 2: Simplified overview of the linking process in mold, highlighting parallel or sequential processing

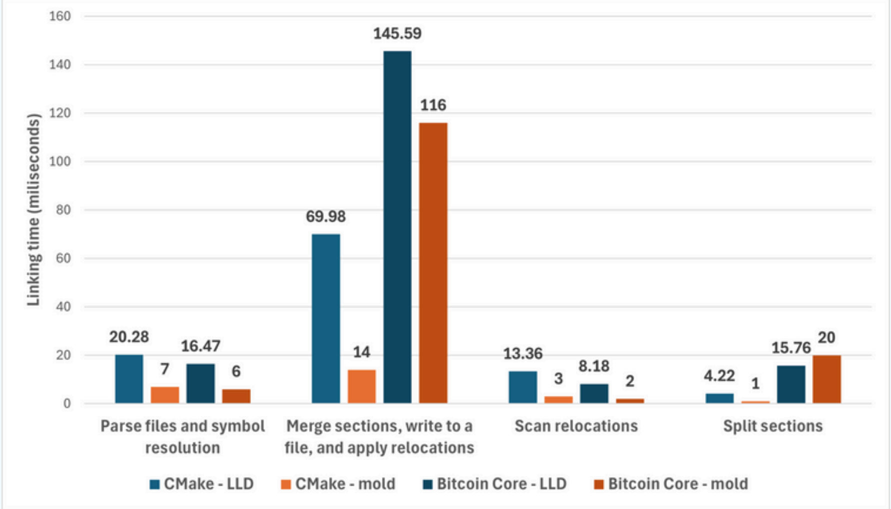


Figure 5: Linking time comparison across different phases between LLD and mold for CMake and Bitcoin Core

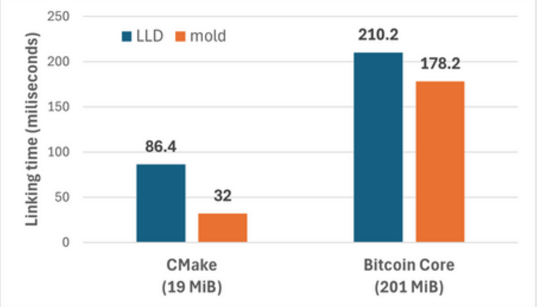


Figure 3: Comparison of linking times between LLD and mold

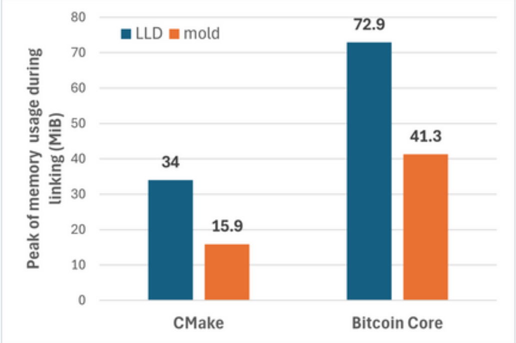
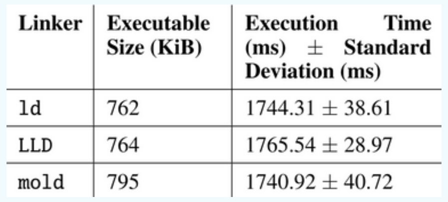


Figure 4: Comparison of memory usage between LLD and mold



Linker	Executable Size (KiB)	Execution Time (ms) ± Standard Deviation (ms)
ld	762	1744.31 ± 38.61
LLD	764	1765.54 ± 28.97
mold	795	1740.92 ± 40.72

Figure 6: Comparison of executables from different linkers for HDiffPatch project, focusing on size, execution times, and section headers

5. DISCUSSION & CONCLUSION

- mold excels in both speed and memory usage.
- the speed advantage of using mold over LLD for linking Bitcoin Core is less pronounced compared to the difference observed when linking CMake - likely due to fewer input files in Bitcoin Core (516) compared to CMake (1143).
- based on belief that implementing linker scripts slows down the linker [3], mold does not support linker scripts → not yet suitable for embedded programming [3].
  - LLD does support linker scripts - their complexity may hinder the implementation of more efficient, parallel processing algorithms - impact remains unclear.
- mold supports only the ELF format, whereas LLD - ELF, PE/COFF, and Mach-O formats.
- mold tends to produce the largest executable.
- execution time of the executable is influenced by the linker:
  - no statistical difference between ld and mold.
  - LLD significantly differs from both ld and mold.
- LLD is slower than mold due to limited parallelization, whereas mold applies extensive parallelization throughout most steps of its process.

[1] D. Thain, "A quick tour," in Introduction to Compilers and Language Design, pp. 5–10, University of Notre Dame, 2nd ed., 2021
[2] LLVM Project, "Linker Design." https://releases.llvm. org/3.8.0/tools/lld/docs/design.html, 2016
[3] R. Ueyama, "Can the mold linker be /usr/bin/ld?," in Proceedings of the FOSDEM Brussels 2024, (Brussels, Belgium), FOSDEM, Feb. 2024