Investigating Build System Parallelism

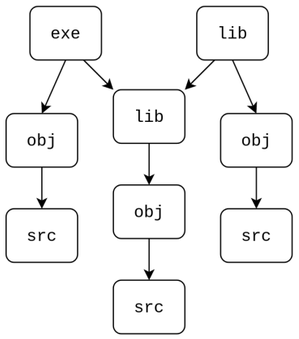
TU Delft

1. Introduction

Shorter compile times mean developers get feedback on their code faster, so they can write more code.

Parallelism improves performance; in C/C++, it is the responsibility of the build system in use. Choosing tasks to execute in parallel is a scheduling problem.

Build systems use a variety scheduling algorithms for this.



It is unknown whether the choice of scheduling affects runtime.

The research question is: **what are the similarities and differences in the task schedulers used by build systems for compiling C/C++ codebases?**

Make [3] and Ninja [1] are popular build systems; experiments also considered Tup [4] and Zig's [2] integrated build system.

Zig's scheduler has access to domain-specific knowledge and should be able to schedule tasks more efficiently.

Hypothesis: Zig will have the fastest runtime.

Ninja uses a smarter scheduling algorithm than Make.

Hypothesis: Ninja will run faster than Make.

This is a dynamic job-shop scheduling problem: a full dependency graph is available but runtimes are unknown.

2. Methodology

The 500 most-installed packages from the Arch Linux repositories were used. They are equipped with configurations for the Arch Build System.

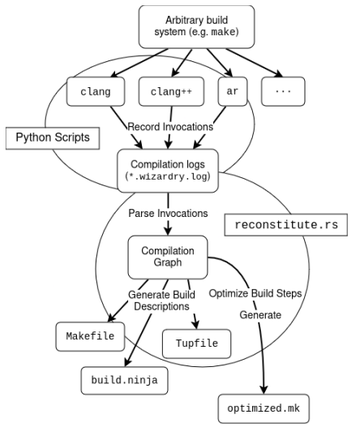
C/C++ codebases from this data set were built and their compiler invocations were recorded.

The command-line parameters of these invocations were parsed and normalized, using data from Zig. Of the 3000+ flags supported by Clang, only 10 were extracted.

A compilation (dependency) graph was constructed and used to produce new build configurations.

Equality-graph saturation was used to simplify the graph and use Zig's integrated build system.

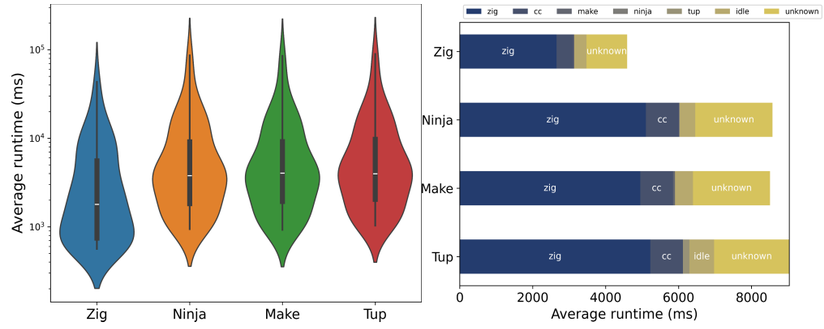
Build systems were profiled in building every codebase, taking data from the OS scheduler.



The collected performance data was processed to determine the total runtime of each build, the amount of CPU time actually spent in each build process, and the amount of CPU idle time. Any uncategorized time was treated as system noise.

Statistical tests (Student's t-test with paired samples) were used to try to refute the hypotheses.

3. Results



Fastest to slowest: Zig, Make, Ninja, Tup
Zig uses parallelism less efficiently than Make

4. Conclusions

The task schedulers used by build systems **do not** significantly affect their runtimes.

Developers should use the Zig build system for C/C++ codebases: it is faster than other compilers and has great features like caching.

Future work should consider developing new compilers with better integrated build systems.

References: [1] github.com/ninja-build/ninja [2] ziglang.org [3] github.com/gittup/tup/ [4] gnu.org/software/make

By Arav Khanna <A.Khanna-1@student.tudelft.nl>
Supervisors: Dennis Sprokholt, Dr. Soham Chakraborty