Compiling code is boring and I don't want to do it

Waiting for the compiler is literally killing me

Átila Neves, Ph.D.

DConf Online 2024

Take away

If it’s noticeable, it’s too slow.

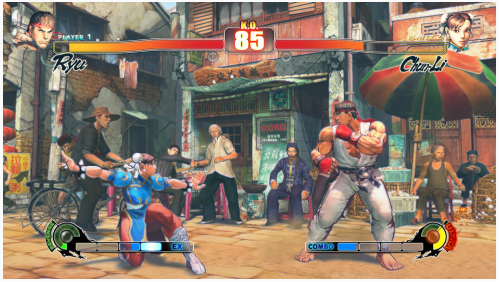
– Átila Neves, 2024

What are the wizards doing?

Turns out, sound is quite slow

- Speed of sound in air at room temperature: 346 m/s
- Latency for 1m: $\sim$ 3 ms
- Latency musicians can tolerate: $\sim$ 10 ms
- Solution: Synchronisio Luminus

What about normies?



- Street Fighter 4 has 1 frame links: $\sim$ 17 ms
- Reflexes are $\sim$ 200 ms
- Reacting isn't the same as anticipating
- 2s is more than noticeable
- 10s to compile is a **long** time for a human

Latency affects productivity

- The longer it takes to get feedback, the longer it takes to progress
- TDD: small change $\rightarrow$ feedback please
- Not TDD? Same thing.
- Productivity is inversely proportional to feedback intervals
- Thesis: compile times are killing productivity

A typical compiler

Buddies!

Why compile anyway?

- Am I currently releasing a binary?
- Do I actually want object files?
- Do I want to pay the "linker tax"?
- Rebuild the world for a 1 line diff??

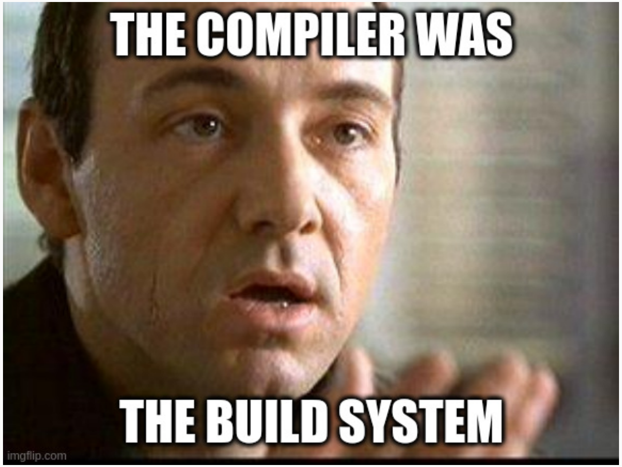
My dream

- “Please run the tests that are impacted”.
- Edited one test? Re-run that test.
- Edited production code? Only run impacted tests.

Incremental compilation — what if. . .

- The compiler were a server
- It only parsed what was strictly necessary
- You got results nearly instantly?

It was the build system this whole time



I wasn't the only one to think of this?

- Examples:
  - Typst
  - Roslyn C# compiler
  - LSP for several languages
- IDE usage optimised for one file being edited
- Inputs: current tree and the diff
- Query system + cache

Backend — JIT Compilation

- No object files
- No linker tax
- No I/O (if the input is from the editor)

Which JIT?

- Many JIT backends
  - libgccjit
  - LLVM JIT
  - GNU Lightning
  - luajit bytecode
  - JVM
- Which is the fastest?
- The fastest at what, exactly?
- Pipeline: edit $\rightarrow$ test result

Experiment with different backends

- Toy language capable of writing a serialisation library
- Parse and bind to different JIT backends
- Benchmark
- Profit?

Ideally

- Fast incremental compilation
- Ability to isolate dependencies
- Fastest way from AST/IR/bytecode to results

Links / References

- Fast typesetting: https://www.user.tu-berlin.de/mhaug/fast-typesetting-incremental-compilation.pdf
- Roslyn design: https://langdev.stackexchange.com/a/2880
- My JIT experiment: https://github.com/atilaneves/jitlang

Once more for those in the back

If it’s noticeable, it’s too slow.

Questions?

Slide intentionally left blank