
emscripten

What is it?

Why Emscripten?

- Legacy codebases
- Programmer productivity
- Cross-platform
- Flexibility

How's it work?

INTERTYPER

- Invoked from compiler.js, given lines of LLVM bitcode as input
- Performs a nearly one-to-one mapping from bitcode instructions to Javascript

ANALYZER

- After Intertyper, invoked by compiler with the intertyper's output
- Gathers type information and data for optimizations
- Returns annotated internal representation

JSIFIER

- Invoked by compiler after analyzer – takes annotated IR as input
- Generates preamble/epilogue boilerplate code and runtime support
- Converts IR into Javascript and outputs it
- Attempts to recreate loops using Relooper to improve performance

The Relooper

- JS engines optimized for “normal” code
  - Loops, if/else, etc – trigger tracing, prediction
- Bitcode only has branch instructions – control flow information is lost
- Given a set of basic blocks, looks at labels, determines reachability, and constructs loops and if/else statements

Runtime

- Native code expects file system, heap, graphics, libraries
- Emscripten provides limited emulation/recreation of these

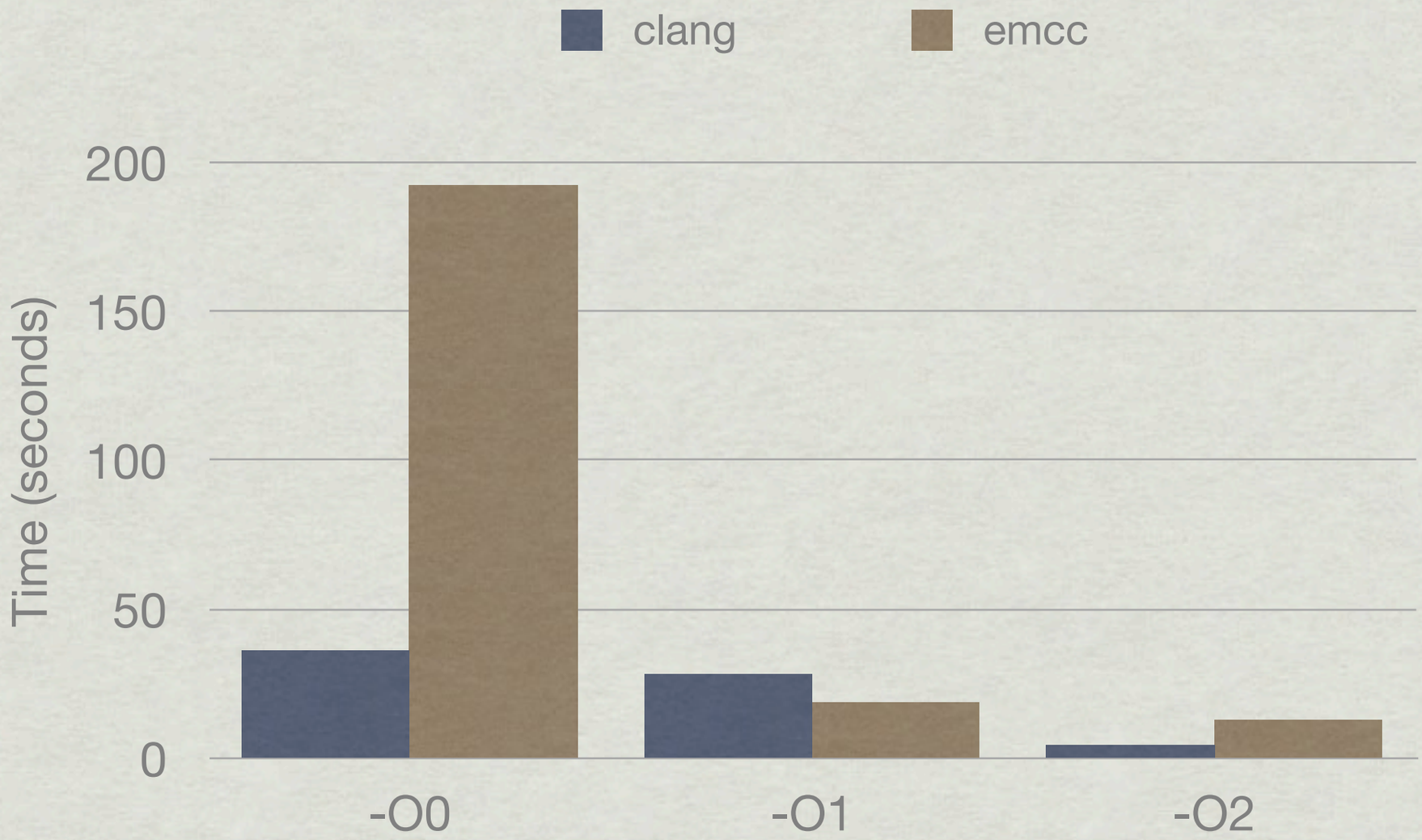
Emscripten FS

- Emulates basic file system to support `fopen` and C standard library functions
- Allows files to be embedded in generated JS
- Also supports Javascript API for file management at runtime

Limitations

- Threads with shared state
- Operations on numeric values
- Low-level hardware-dependent instructions

Performance



Demo

Questions?