Not So Fast: Analyzing the Performance of WebAssembly vs. Native Code

Abhinav Jangda, Bobby Powers, Emery Berger, Arjun Guha

University of Massachusetts Amherst
WebAssembly

WebAssembly is becoming popular
Is WebAssembly fast?

WebAssembly evaluation on PolyBenchC Benchmarks (Haas et al. PLDI 2017)

WebAssembly is 17% faster than asm.js   WebAssembly is 26% slower than native
PolyBenchC Benchmarks are **not** representative of WebAssembly use cases

Scientific Applications

PolyBenchC benchmarks are scientific kernels

Image/Video Processing

Compilers, debuggers, interpreters, and virtual machines

POSIX Applications
SPEC CPU benchmarks are better representative of WebAssembly use cases

1. 433.milc
2. 444.namd
3. 447.dealII
4. 450.soplex
5. 470.lbm
6. 644.nab_s
7. 429.mcf
8. 462.libquantum

1. 464.h264ref
2. 453.povray
1. 400.perlbench
2. 403.gcc

1. 401.bzip2
2. 429.mcf
3. 433.milc
4. 444.namd
5. 445.gobmk
6. 450.soplex
7. 453.povray
8. 458.sjeng
9. 462.libquantum
10. 470.lbm
11. 473.astar
12. 482.sphinx3
13. 631.leela_s
14. 641.nab_s
Cannot execute SPEC benchmarks in browsers

System Calls and Filesystem

No system call support for Web Applications
Browsix-WASM provides system calls for WebAssembly apps
Browsix-WASM overhead is minimal

Maximum overhead is 1.1%.
Average overhead is 0.2%.
Performance of SPEC benchmarks in WebAssembly

WebAssembly: 30% faster than asm.js

WebAssembly: 55% slower than native
Why is there a gap?

1. Poor register allocation
2. Reserved registers
3. Poor instruction selection
4. Stack overflow checks
5. Indirect function call checks
Not So Fast: Analyzing the Performance of WebAssembly vs. Native Code

Abhinav Jangda, Bobby Powers, Emery Berger, Arjun Guha
University of Massachusetts Amherst

ATC 2019
Wednesday, July 10
12:20 - 12:40 PM in Runtime Track