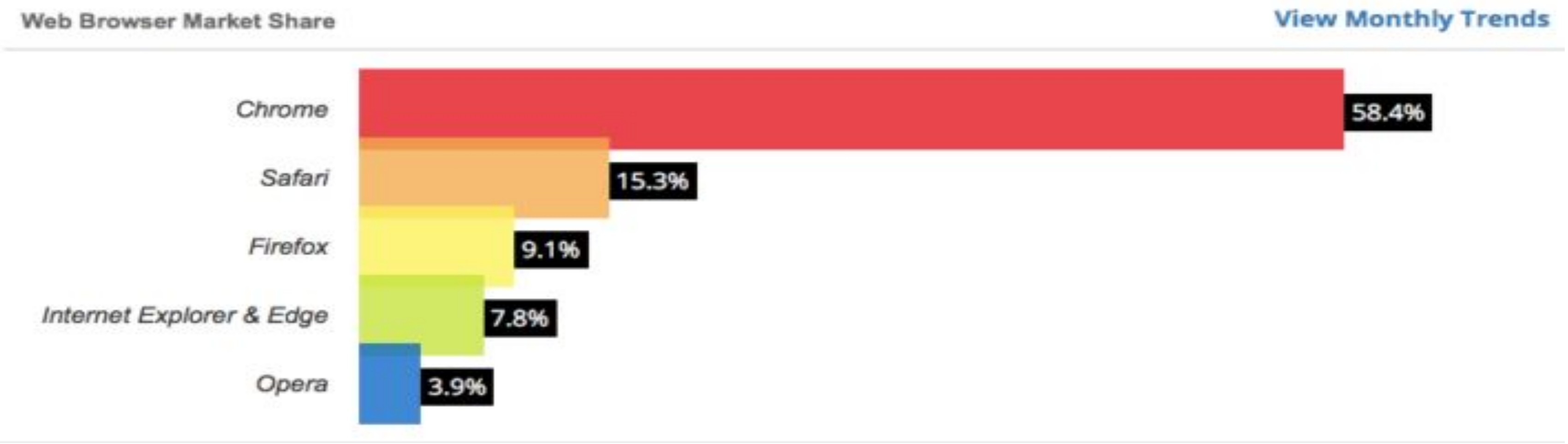
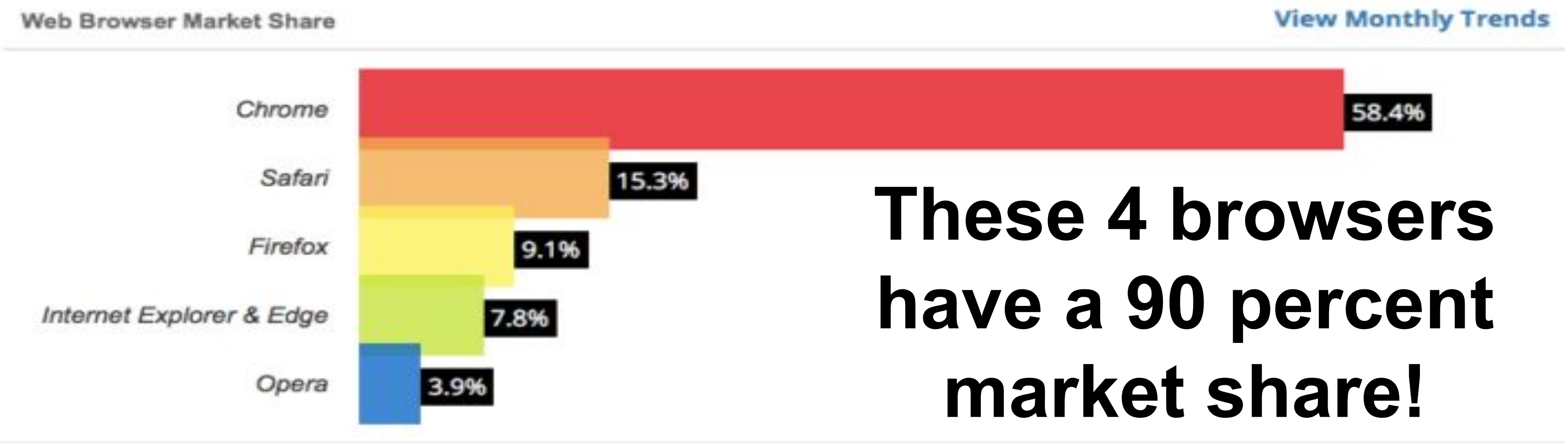
Bringing the Web Up to Speed with WebAssembly

Matthew Furlong, Drew Davis, Ayush Goel, Umang Lathia

An Open Standard



An Open Standard



These 4 browsers have a 90 percent market share!

Outline

1. **Introduction and Motivation**
2. Overview and Execution
3. Validation
4. Binary Format & Embedding
5. Evaluation

The Web

- “The most ubiquitous application platform ever.”

The Web

- “The most ubiquitous application platform ever.”
- Yet Javascript is the only natively supported programming language on the web...

```
> typeof NaN
< "number"
> 9999999999999999
< 10000000000000000
> 0.5+0.1==0.6
< true
> 0.1+0.2==0.3
< false
> Math.max()
< -Infinity
> Math.min()
< Infinity
> []+[]
< ""
> []+{}
< "[object Object]"
> {}+[]
< 0
> true+true+true===3
< true
> true-true
< 0
```

```
> true==1
< true
> true===1
< false
> (!+[]+[]+![]).length
< 9
> 9+"1"
< "91"
> 91-"1"
< 90
> []==0
< true
```

Away from Javascript...

- Web applications are more demanding than ever
  - 3D Visualization
  - Audio and Video software
  - Games
- Many developers don't want to use Javascript

... and Onto WebAssembly!

- A low-level, language independent bytecode for the Web

... and Onto WebAssembly!

- A low-level, language independent bytecode for the Web
- Goals
    - Safe
    - Fast
    - Portable
    - Compact

Previous Work on Bytecode for the Web

- Microsoft's ActiveX
- Native Client and Portable Native Client
- asm.js

Previous Work on Bytecode for the Web

- Microsoft's ActiveX
- Native Client and Portable Native Client
- asm.js

WebAssembly is the first solution for low-level code on the Web that provides safety, speed, portability, and small code sizes.

Outline

1. Introduction and Motivation
2. **Overview and Execution**
3. Validation
4. Binary Format & Embedding
5. Evaluation

Overview

- A binary code format, not a language
- Basic language features
  - Modules
  - Functions
  - Instructions
  - Traps
  - ....

Overview

- New Language features
  - Linear memory (also known as flat memory)
  - Endiannes
    - Little endian
  - Structured Control Flow
    - Eliminates problems caused by simply jumps
    - Blocks execute like function calls
  - Function calls

Overview

- Determinism
  - Design semantics tries to minimize non determinism due to corner cases.
  - Implementation dependent behavior
    - NaNs
    - Resource Exhaustion
    - Host Functions

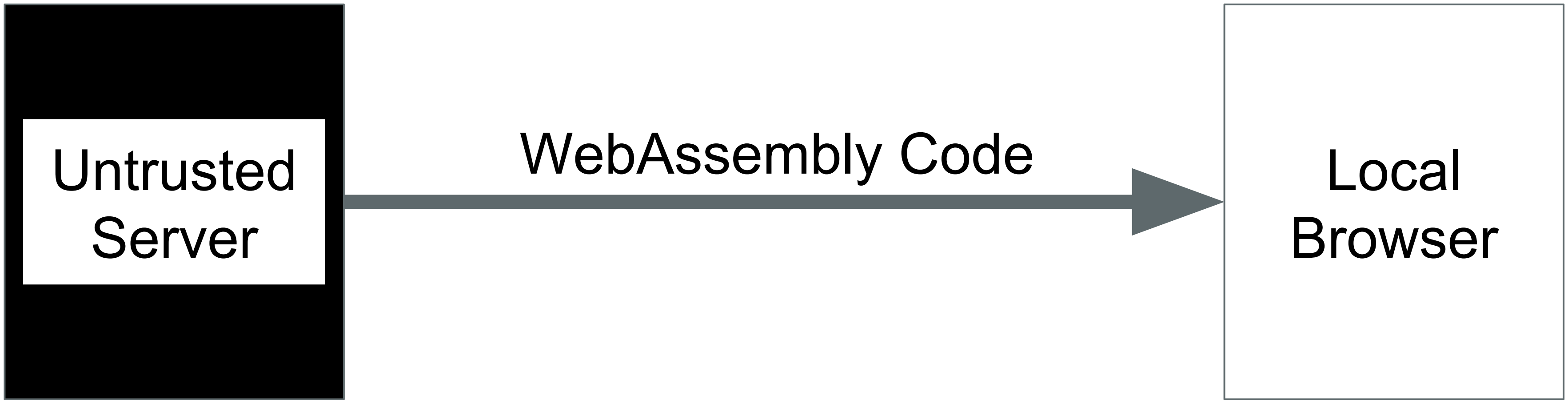
Execution

- Uses a global store object (like Windows in Browsers)
- Stores and Runtime objects representation
- Reduction Rules

Outline

1. Introduction and Motivation
2. Overview and Execution
3. **Validation**
4. Binary Format & Embedding
5. Evaluation

Validation

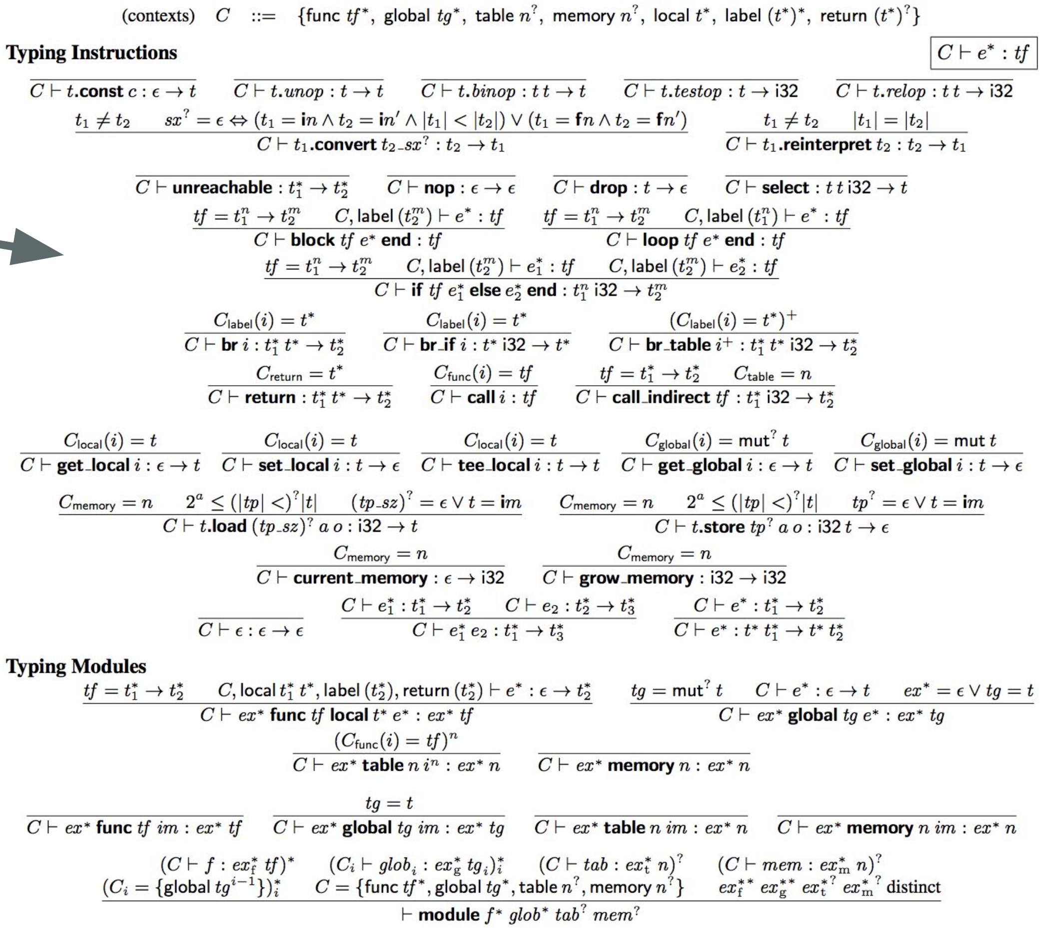


Validation

- Defined as a simple *type system*
- Efficiently checkable in a single linear pass

Validation

- Typing Rules
  - Ensure that the typ
    correct
- Soundness
  - Typing rules cover
    behavior)
    - Guarantees me
      code addresses



Outline

1. Introduction and Motivation
2. Overview and Execution
3. Validation
4. **Binary Format & Embedding**
5. Evaluation

Binary Format

- Code transmitted across web as a binary encoding
  - Binary code organized by entities
    - Streaming compilation
    - Parallelized compilation
  - Instructions - one-byte opcodes
  - Integral numbers - LEB128 format

Embedding

- WebAssembly is designed to be embedded into an execution environment
- Therefore, does not define:
    - How programs are loaded into execution environment
    - How I/O is performed

Outline

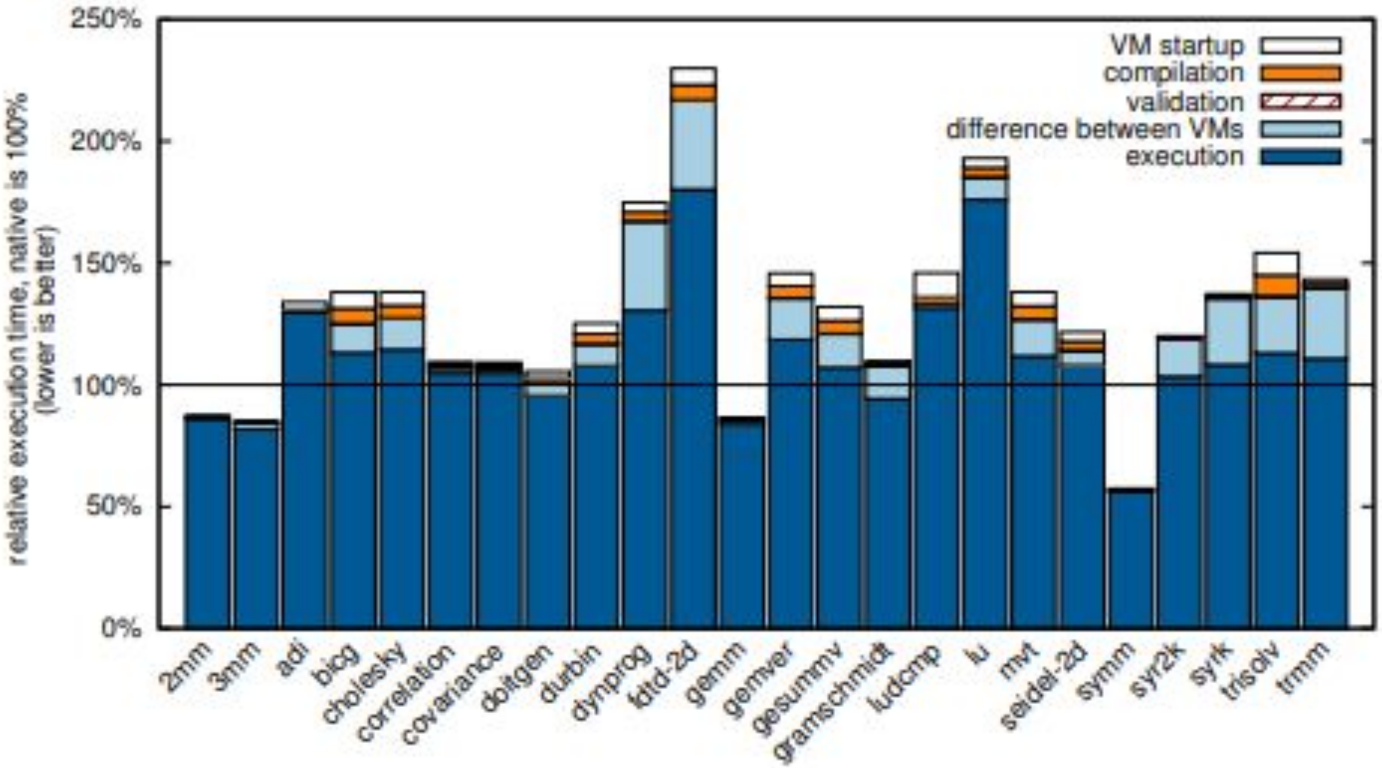
1. Introduction and Motivation
2. Overview and Execution
3. Validation
4. Binary Format & Embedding
5. **Evaluation**

Implementation

- Lots of different JavaScript engines
  - V8 (Chrome), SpiderMonkey (Mozilla), Chakra (Edge)
- Developed independent implementations for each browser
  - On-the-fly validation (as fast as 1 GB/s)
  - SSA (V8 and SpiderMonkey) → direct-to-SSA in a single pass
- Other Optimizations
  - Bounds Check - Constant-fold memsize - offset
  - Parallel Compilation (5-6x improvement)
  - Compiled code caching (memoization)

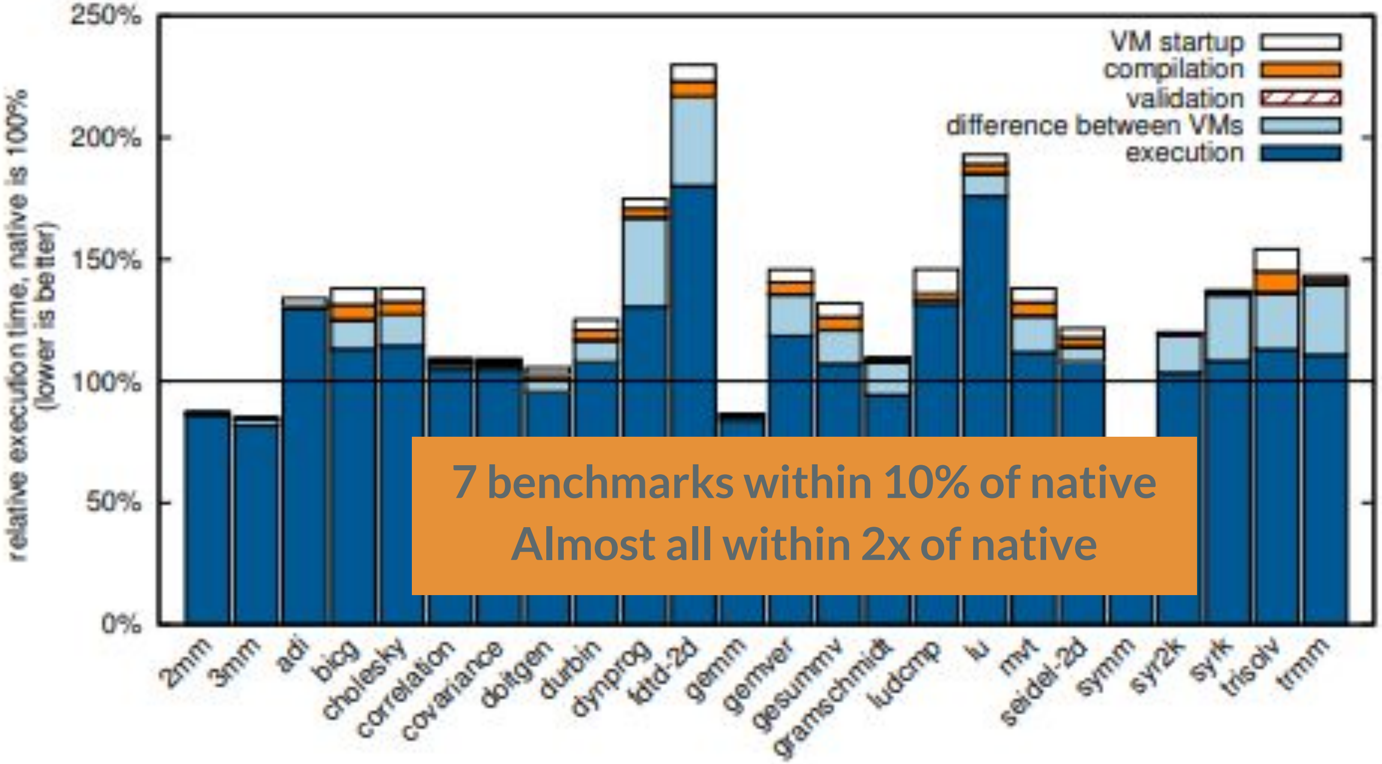
Measurements

Execution time of PolyBenchC benchmarks on Webassembly normalized to native code



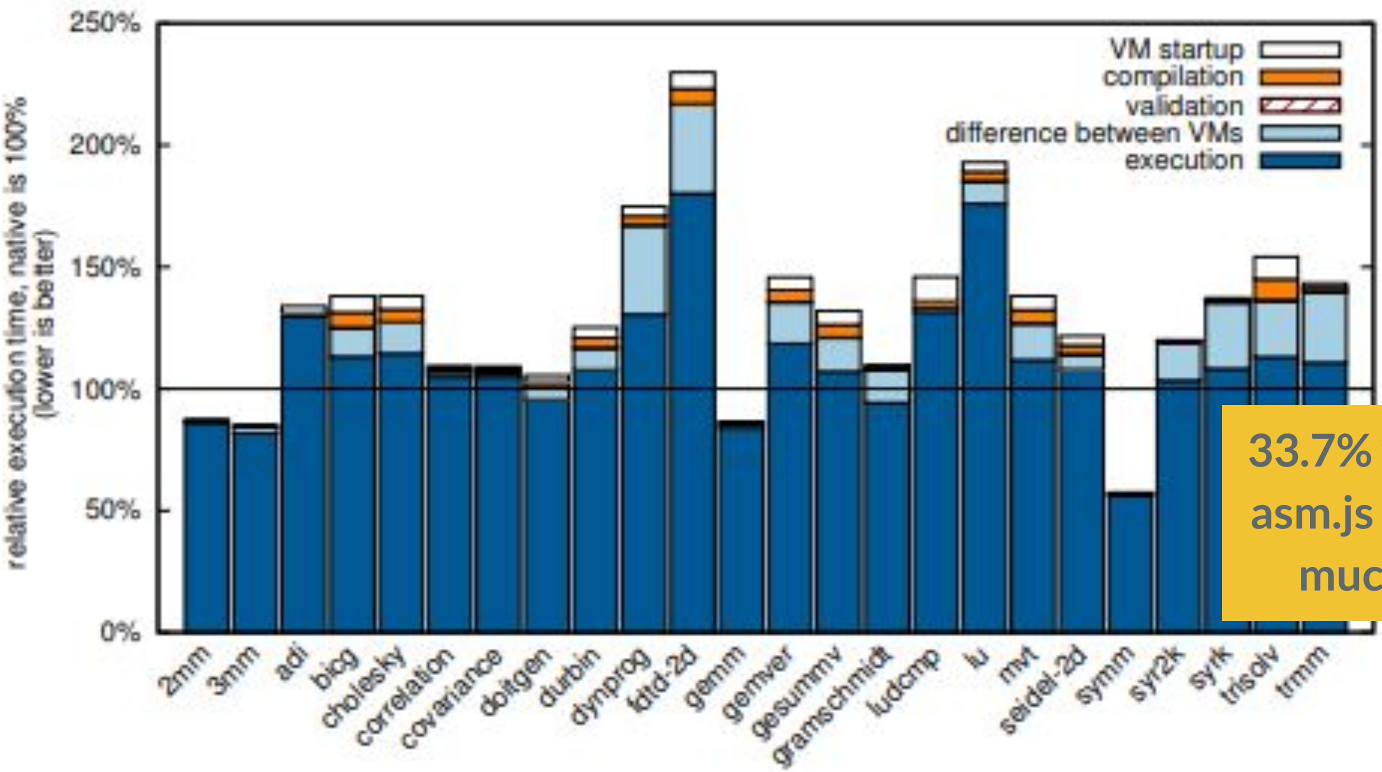
Measurements

Execution time of PolyBenchC benchmarks on Webassembly normalized to native code



Measurements

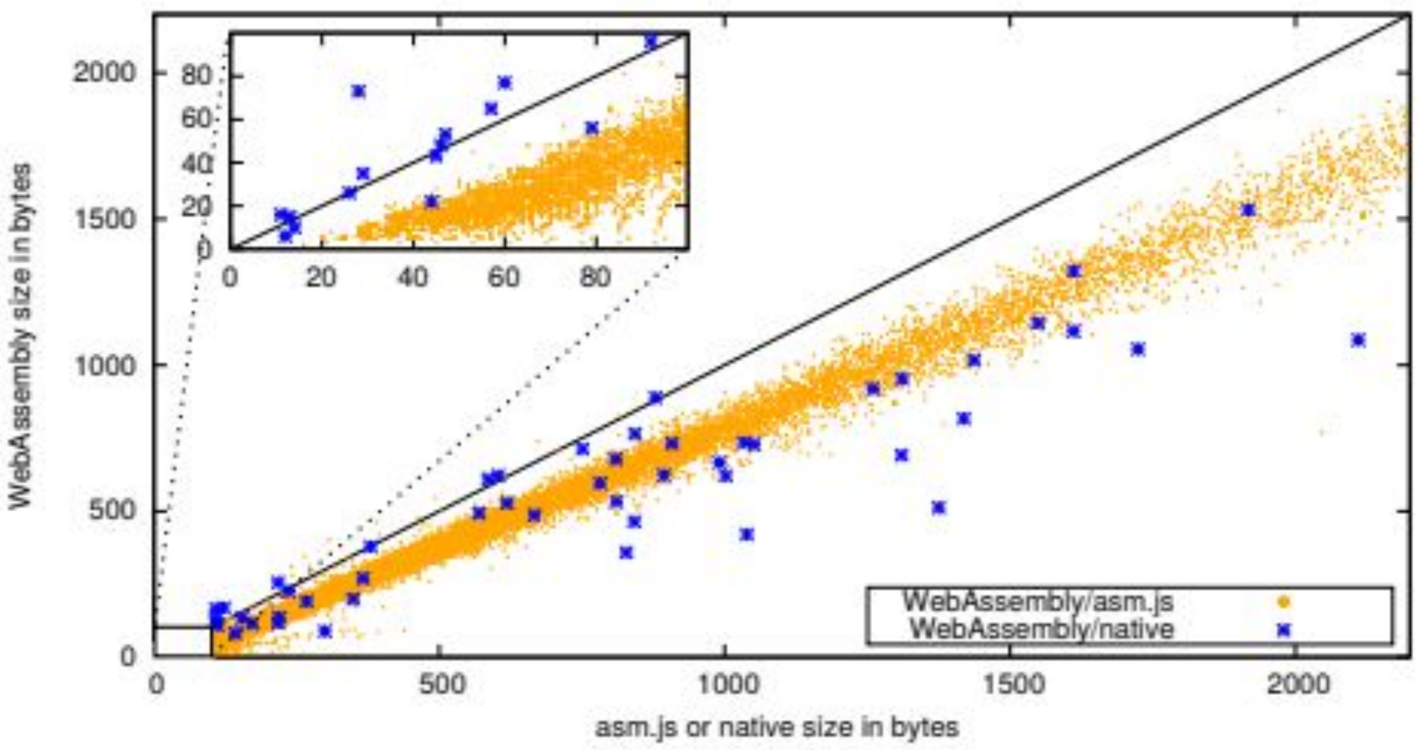
Execution time of PolyBenchC benchmarks on Webassembly normalized to native code



33.7% faster than asm.js (validation much faster)

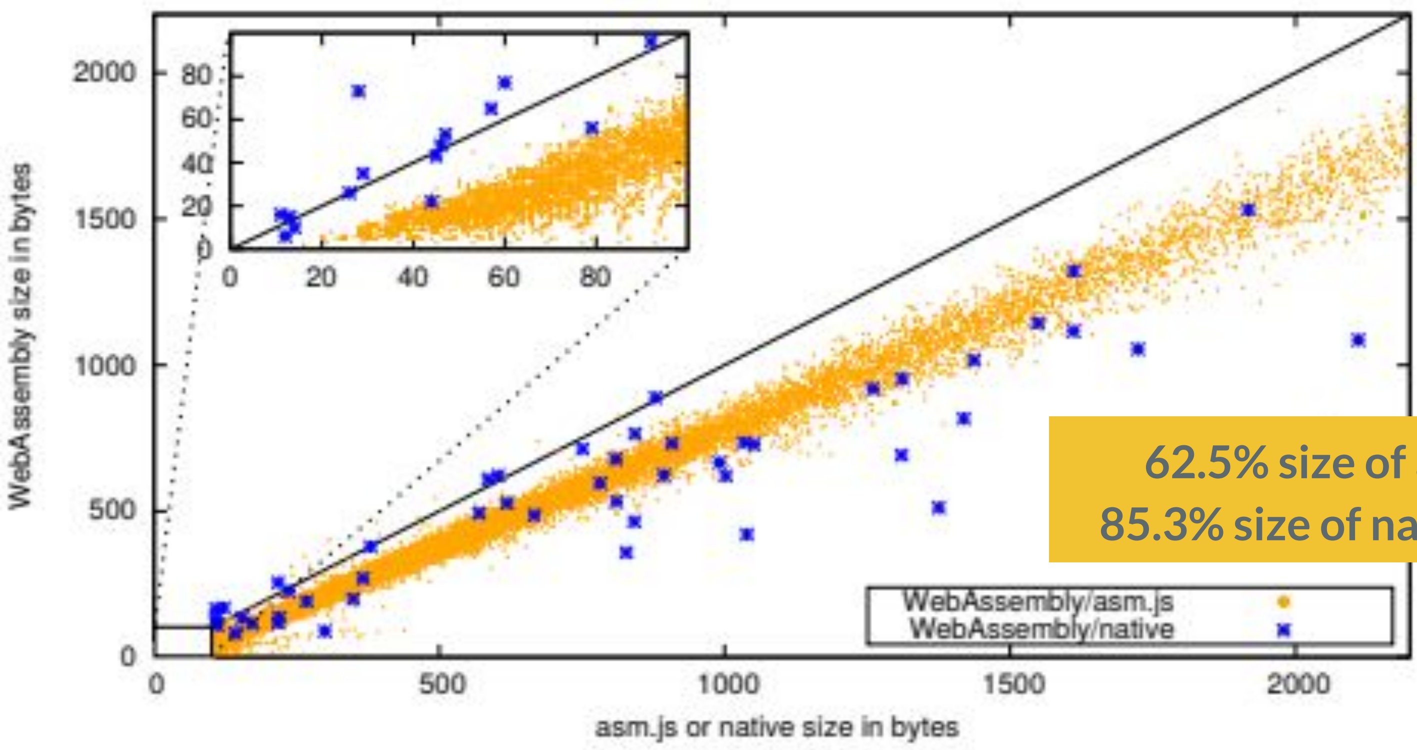
Measurements

Binary size of WebAssembly in comparison to asm.js and native code



Measurements

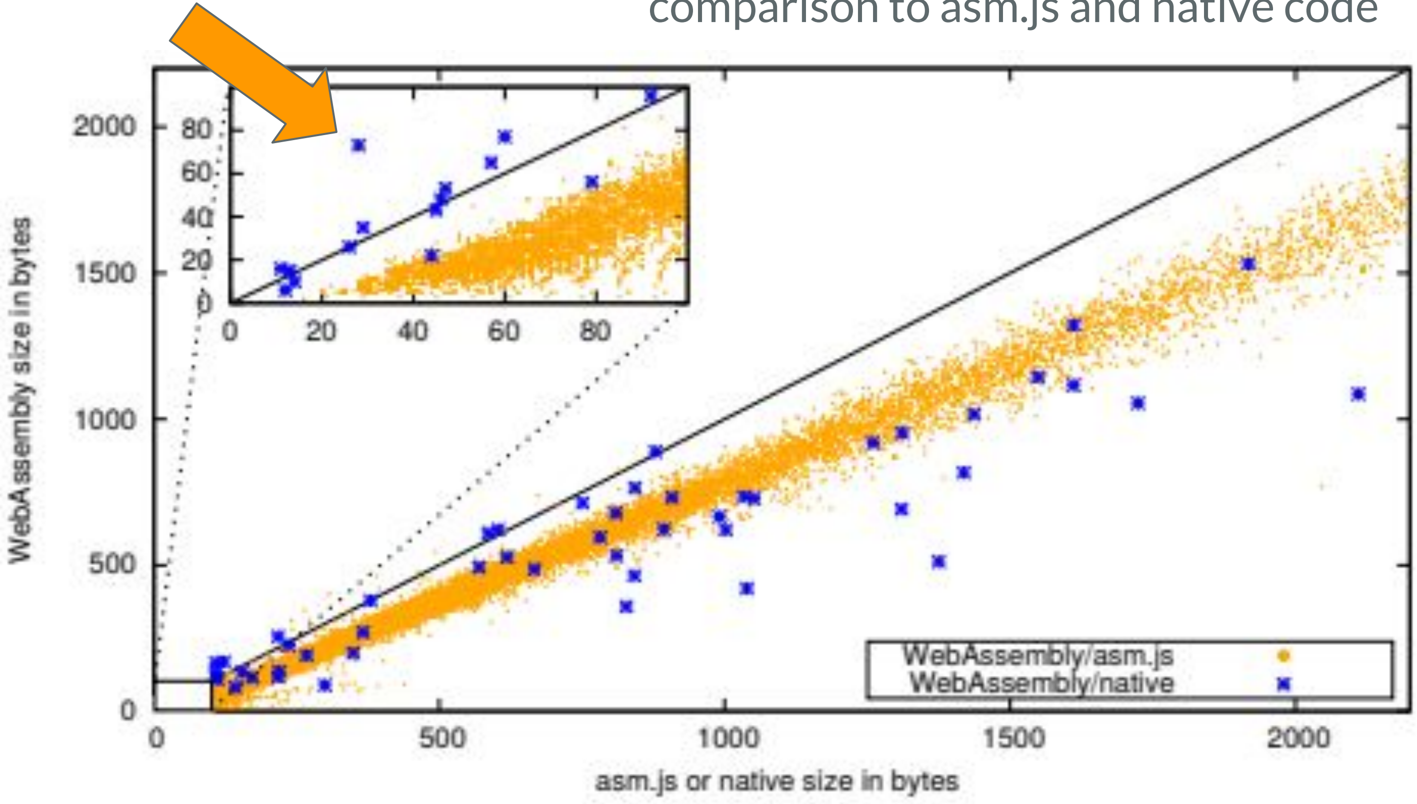
Binary size of WebAssembly in comparison to asm.js and native code



62.5% size of asm.js
85.3% size of native x86

Measurements

Binary size of WebAssembly in comparison to asm.js and native code



Evaluation

- Strength
    - Ability to write in any language
    - Faster compilation
    - Compact
    - Fast
- Weaknesses
    - Separate compiler to port each language to WebAssembly

Road Map

- MVP Completed (3 years ago)
- Features in process:
  - Exception handling
  - Threads
  - Garbage Collection
  - Single Instruction Multiple Data instructions
  - Tail Calls

https://webassembly.org/docs/future-features/

Community & Current Updates

Understand WebAssembly: Why It Will Change the Web	Dec 19, 2017 -
Why WebAssembly is a game changer for the web—and a source of pride for Mozilla and Firefox	Mar 7, 2017 -
Rust language gets direct WebAssembly compilation	Nov 29, 2017
Making WebAssembly even faster: Firefox's new streaming and tiering compiler	Jan 17, 2018 -
WebAssembly Threads ready to try in Chrome 70	Nov 5, 2018 -

Questions?

Appendix A - Why is WebAssembly faster than asm.js?

- Startup
    - Smaller to download, faster to parse
- CPU features
    - asm.js doesn't have access to CPU features -- slower
-