WasmScore

Wasm Benchmarking CG Subgroup

Johnnie Birch

3/8/2024

The Motivation

The questions from those who aren’t necessarily Wasm runtime developers or enthusiasts:

- Can you point me to a standard Webassembly benchmark? Is there something like a SpecJBB equivalent for Wasm that we can use to track performance?
- In elevator pitch terms, tell me how well does Wasm perform on platform X compared to platform Y compared to platform Z?
- You report that your Wasm benchmark ran in N seconds .. is this good or bad?
- Answer me, do certain categories of codes run better in Wasm than others?
- You are convincing me to target Wasm by saying it has near native performance, but how near is “near”? If my workload is compiled to Wasm instead of native, will there be a noticeable drop in performance?

The Requirements

The requirements of a benchmark to answer the previous questions:

- That provides a simple distilled assessment of the underlying platform ability to support Wasm.
- Provides the user with a baseline to gauge the Wasm performance they see.
- That is portable and that produces comparable scores when run on different platforms.
- Is easy to get started. Does not require a laundry list of tools to preinstall and does not require a build step.
- Provides results that are easy to digest. Does not require someone to even understand what Wasm is to be able to see and compare performance.
- Gives you valuable results. Underlying results are repeatable, relevant, and respected.

Why the Design

Why Docker?

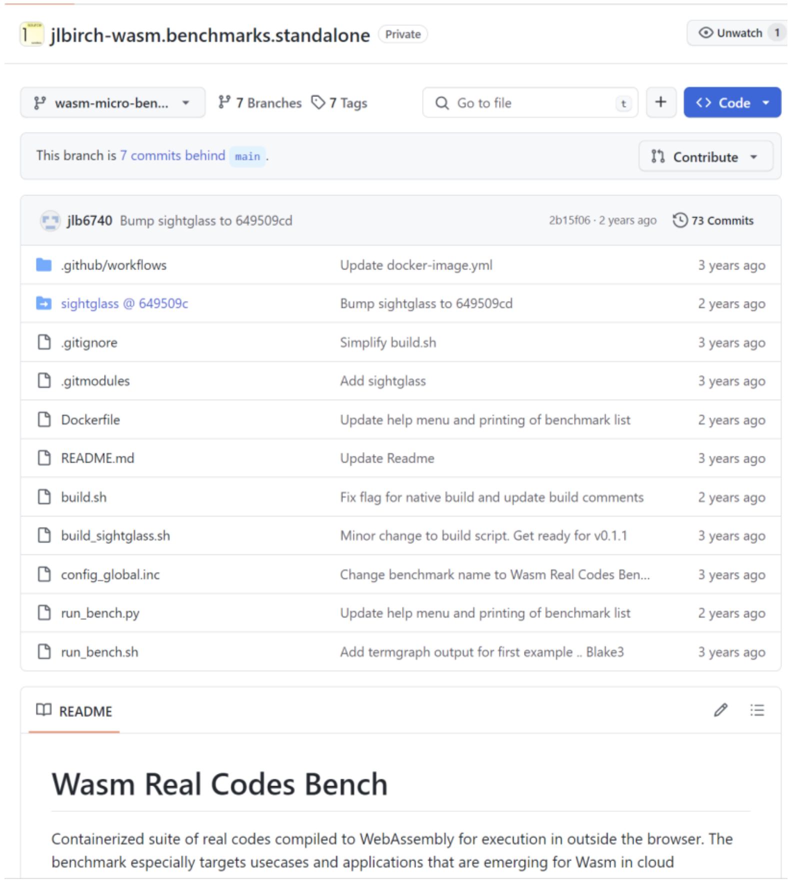
- Lighting fast to get started
- Cross platform
- Influenced by the convenience of getting performance assessments using Android Apps.

Why Suites?

- Again, influenced by Apps that packaged assessments of the CPU, Graphics subsystem, memory, etc into one convenient app and defined each assessment by a suite of benchmarks.

Why Sightglass as the underlying driver?

- Have been leveraging sighglass since 1.0 so I was a user, contributor, and it was very familiar.
- It is the perfect candidate: Well, thought out, trusted, and has active support.


jlbirch-wasm.benchmarks.standalone Private

This branch is 7 commits behind main.

jlb6740 Bump sightglass to 649509cd 2b15f06 · 2 years ago 73 Commits

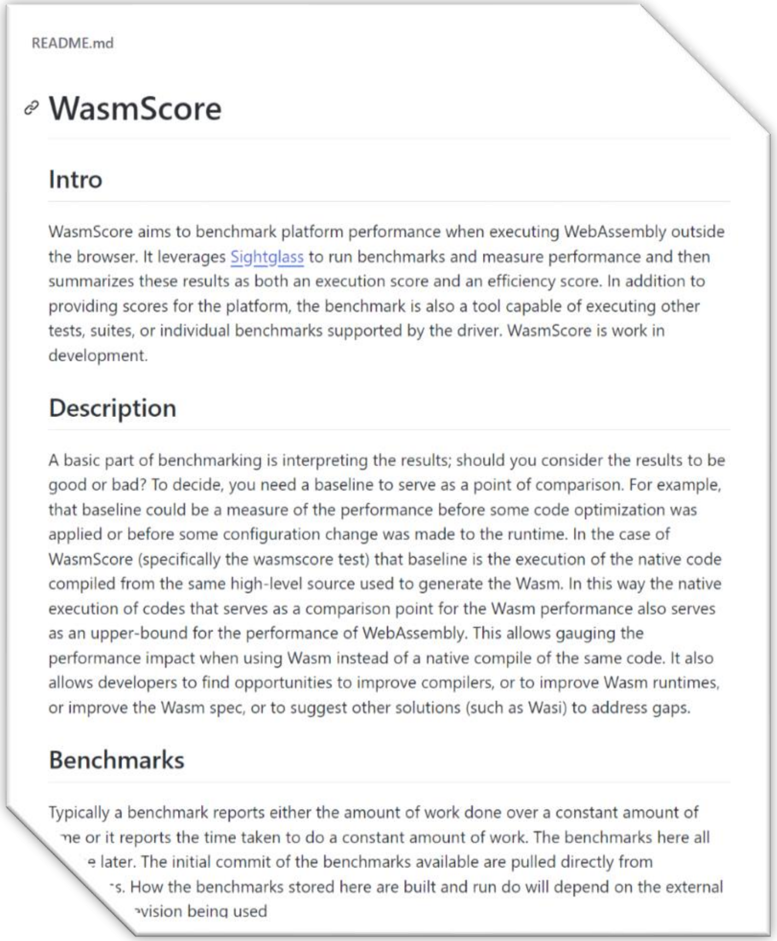
.github/workflows	Update docker-image.yml	3 years ago
sightglass @ 649509c	Bump sightglass to 649509cd	2 years ago
.gitignore	Simplify build.sh	3 years ago
.gitmodules	Add sightglass	3 years ago
Dockerfile	Update help menu and printing of benchmark list	2 years ago
README.md	Update Readme	3 years ago
build.sh	Fix flag for native build and update build comments	2 years ago
build_sightglass.sh	Minor change to build script. Get ready for v0.1.1	3 years ago
config_global.inc	Change benchmark name to Wasm Real Codes Ben...	3 years ago
run_bench.py	Update help menu and printing of benchmark list	2 years ago
run_bench.sh	Add termgraph output for first example .. Blake3	3 years ago

README

Wasm Real Codes Bench

Containerized suite of real codes compiled to WebAssembly for execution in outside the browser. The benchmark especially targets usecases and applications that are emerging for Wasm in cloud

Github Repo and Demo



README.md

WasmScore

Intro

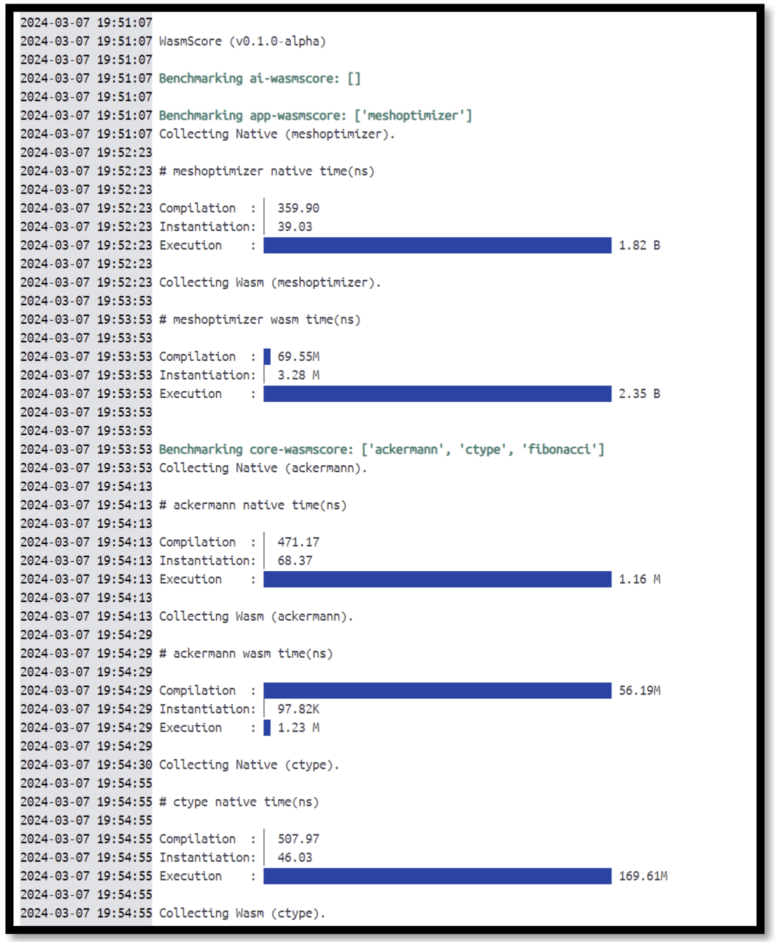
WasmScore aims to benchmark platform performance when executing WebAssembly outside the browser. It leverages Sightglass to run benchmarks and measure performance and then summarizes these results as both an execution score and an efficiency score. In addition to providing scores for the platform, the benchmark is also a tool capable of executing other tests, suites, or individual benchmarks supported by the driver. WasmScore is work in development.

Description

A basic part of benchmarking is interpreting the results; should you consider the results to be good or bad? To decide, you need a baseline to serve as a point of comparison. For example, that baseline could be a measure of the performance before some code optimization was applied or before some configuration change was made to the runtime. In the case of WasmScore (specifically the wasmscore test) that baseline is the execution of the native code compiled from the same high-level source used to generate the Wasm. In this way the native execution of codes that serves as a comparison point for the Wasm performance also serves as an upper-bound for the performance of WebAssembly. This allows gauging the performance impact when using Wasm instead of a native compile of the same code. It also allows developers to find opportunities to improve compilers, or to improve Wasm runtimes, or improve the Wasm spec, or to suggest other solutions (such as Wasi) to address gaps.

Benchmarks

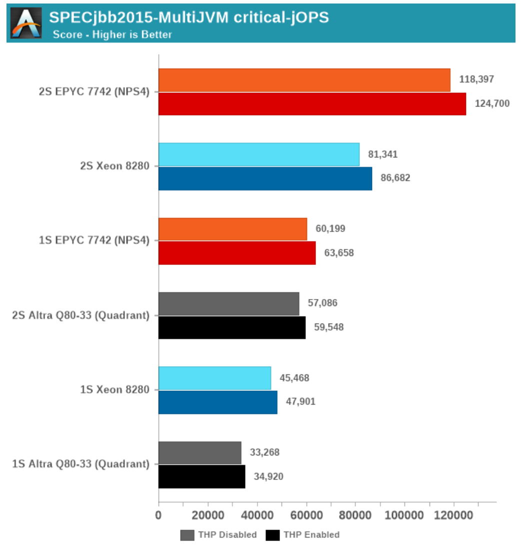
Typically a benchmark reports either the amount of work done over a constant amount of
me or it reports the time taken to do a constant amount of work. The benchmarks here all
e later. The initial commit of the benchmarks available are pulled directly from
s. How the benchmarks stored here are built and run do will depend on the external
vision being used



```
2024-03-07 19:51:07
2024-03-07 19:51:07 WasmScore (v0.1.0-alpha)
2024-03-07 19:51:07
2024-03-07 19:51:07 Benchmarking ai-wasmscore: []
2024-03-07 19:51:07
2024-03-07 19:51:07 Benchmarking app-wasmscore: ['meshoptimizer']
2024-03-07 19:51:07 Collecting Native (meshoptimizer).
2024-03-07 19:52:23
2024-03-07 19:52:23 # meshoptimizer native time(ns)
2024-03-07 19:52:23
2024-03-07 19:52:23 Compilation  : │ 359.90
2024-03-07 19:52:23 Instantiation: │ 39.03
2024-03-07 19:52:23 Execution    : ██████████████████████████████████████████████ 1.82 B
2024-03-07 19:52:23
2024-03-07 19:52:23 Collecting Wasm (meshoptimizer).
2024-03-07 19:53:53
2024-03-07 19:53:53 # meshoptimizer wasm time(ns)
2024-03-07 19:53:53
2024-03-07 19:53:53 Compilation  : ▌ 69.55M
2024-03-07 19:53:53 Instantiation: │ 3.28 M
2024-03-07 19:53:53 Execution    : ██████████████████████████████████████████████ 2.35 B
2024-03-07 19:53:53
2024-03-07 19:53:53
2024-03-07 19:53:53 Benchmarking core-wasmscore: ['ackermann', 'ctype', 'fibonacci']
2024-03-07 19:53:53 Collecting Native (ackermann).
2024-03-07 19:54:13
2024-03-07 19:54:13 # ackermann native time(ns)
2024-03-07 19:54:13
2024-03-07 19:54:13 Compilation  : │ 471.17
2024-03-07 19:54:13 Instantiation: │ 68.37
2024-03-07 19:54:13 Execution    : ██████████████████████████████████████████████ 1.16 M
2024-03-07 19:54:13
2024-03-07 19:54:13 Collecting Wasm (ackermann).
2024-03-07 19:54:29
2024-03-07 19:54:29 # ackermann wasm time(ns)
2024-03-07 19:54:29
2024-03-07 19:54:29 Compilation  : ██████████████████████████████████████████████ 56.19M
2024-03-07 19:54:29 Instantiation: │ 97.82K
2024-03-07 19:54:29 Execution    : ▌ 1.23 M
2024-03-07 19:54:29
2024-03-07 19:54:30 Collecting Native (ctype).
2024-03-07 19:54:55
2024-03-07 19:54:55 # ctype native time(ns)
2024-03-07 19:54:55
2024-03-07 19:54:55 Compilation  : │ 507.97
2024-03-07 19:54:55 Instantiation: │ 46.03
2024-03-07 19:54:55 Execution    : ██████████████████████████████████████████████ 169.61M
2024-03-07 19:54:55
2024-03-07 19:54:55 Collecting Wasm (ctype).
```

Why Performance Scores?

- Scores are how all benchmarks report. Any metric can be called/considered a score.
- In WasmScore scores are intended be nothing more than summaries of aggregated metrics.
  - Performance: 1/(Geomean Execution Time)
  - Efficiency: (Time Wasm/ Time Native)
  - TODO: SIMD Score, WASI Score



Antutu
New Model Exposure

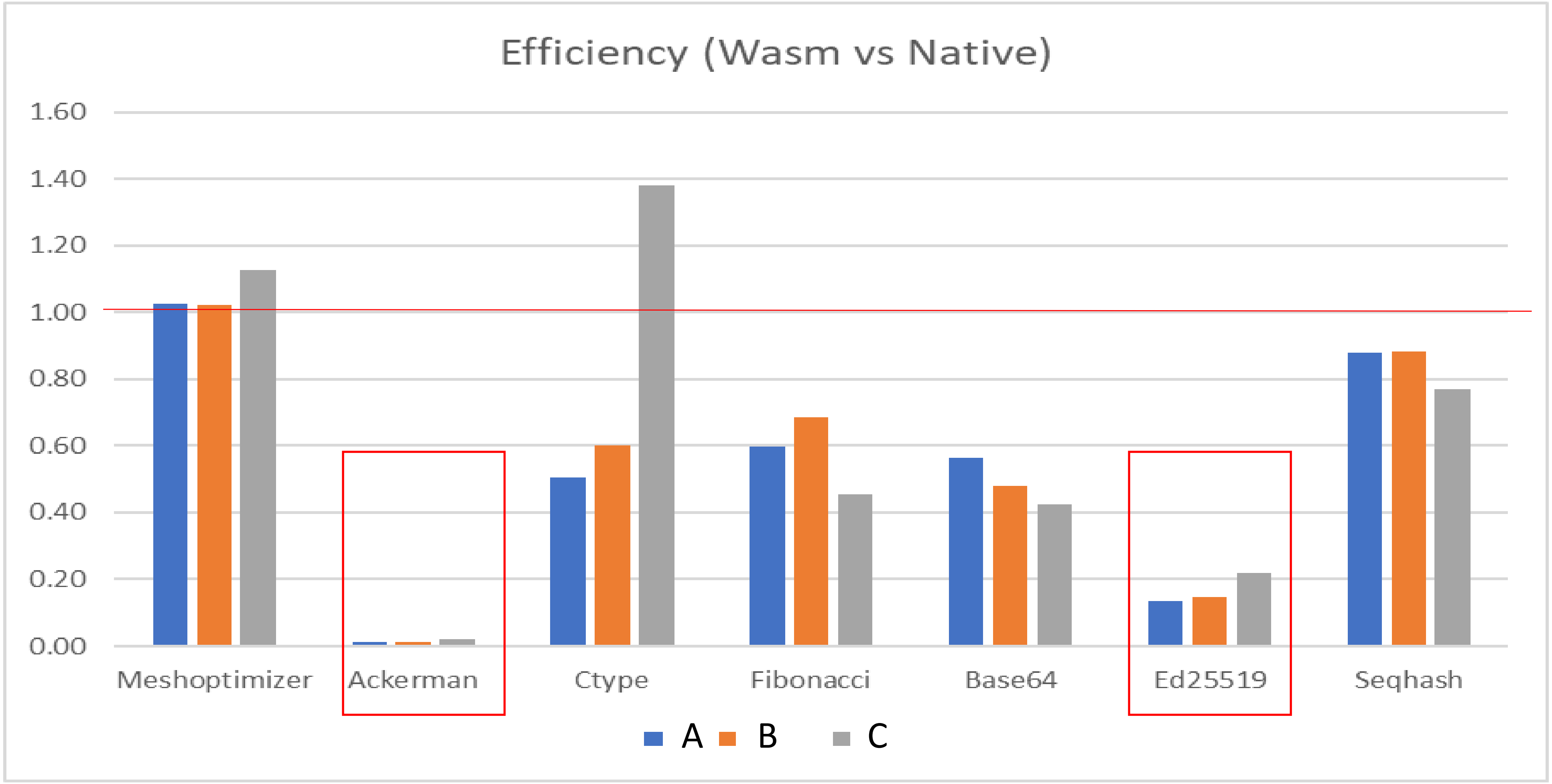
Name: iPhone SE (2020) Model: iPhone 12,8
Configuration: 3GB+256GB System: iOS 13.4

492166分
(SoC: Apple A13)

CPU	149295
GPU	186079
MEM	77968
UX	78824

WasmScore – Efficiency

Confidential/Invalid .. Not to be shared or used



- Red line shows parity in performance with native.

Summary

Goals of Ease of use and User experience

- Simple to download, install and run.
- Wants you to not need to be a developer with exclusive knowledge to interpret the results communicated.
- A goal is to aggregate data in a way that is convenient for charting and searching for anomalies

Leverages Sightglass directly (Not a fork)

- Sightglass is well a thoughtout and well written benchmark tool and it has community interest
- Certain features Sightglass may want (such as support for a native engine).
- Other aspect such as the workload repository and definitions for suites remain separate to avoid having Sightglass be a dependency there.

Sightglass collects the data, but it was not the goal of WasmScore to be a UI for Sightglass

- It's about the aggregation of Wasm performance data into a few trackable, comparable scores
- It's about the convenience of installing, collecting, and post processing the results.
- It's about the motivation of assessing and comparing the performance of the underlying platform.