Code Speed Tracking Infrastructure for compilers, runtimes, libs

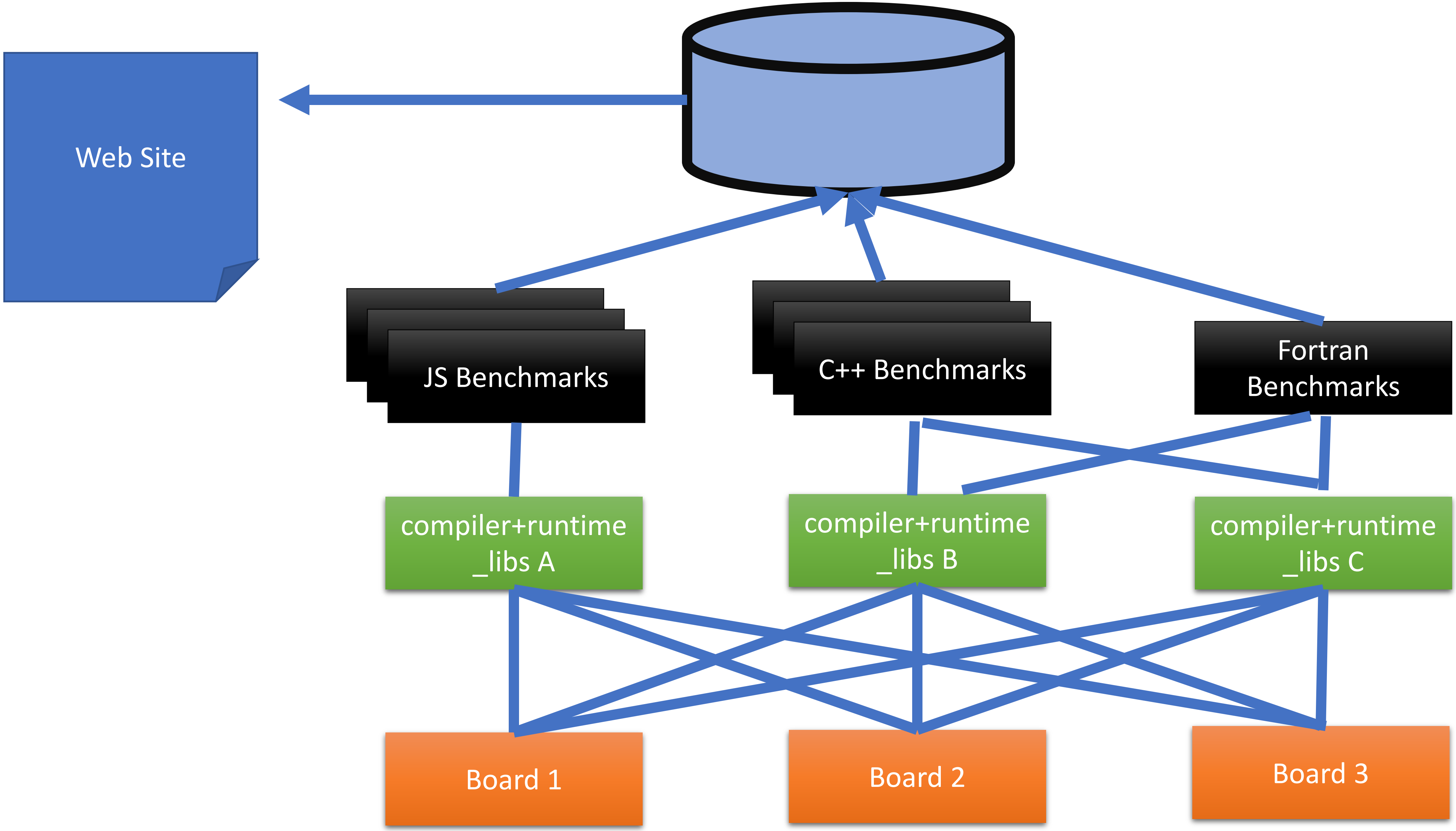
RFC Proposal, first version

Wei Wu lazyparser@gmail.com

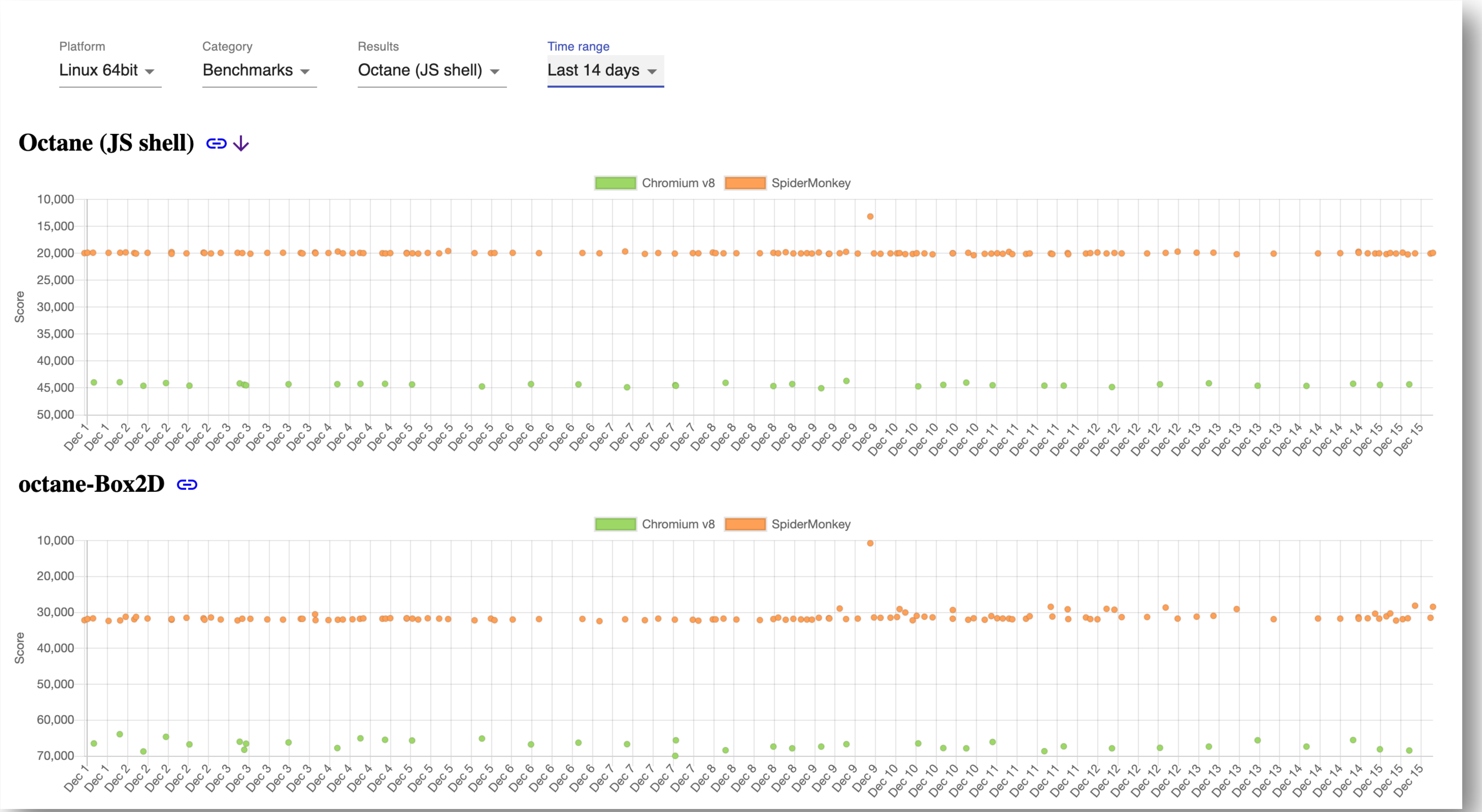
2020-12-15

Overview - scope

- Build a performance tracking system for RISC-V ecosystem
  - Running on physical boards. Get real performance data.
    - Hifive Unleashed is ready. Hifive Unmatched and PolarFire icicle will follow.
  - Targeting GCC, Clang/LLVM, V8, Spidermonkey, OpenJDK, Rust, Golang, and LuaJIT, etc.
  - Also targeting OpenBLAS and other important libraries in future
  - Using Both free and commerical Benchmarks
    - Free: mBench? SunSpider, Octane, Kraken
    - Non-free: SPEC CPU 2006, etc.
  - A website to show the data points. Using arewefastyet open-source project


Web Site
JS Benchmarks
C++ Benchmarks
Fortran Benchmarks
compiler+runtime_libs A
compiler+runtime_libs B
compiler+runtime_libs C
Board 1
Board 2
Board 3

Demo - https://arewefastyet.com/linux64/benchmarks/octane?numDays=60



Platform: Linux 64bit

Category: Benchmarks

Results: Octane (JS shell)

Time range: Last 14 days

Octane (JS shell)

octane-Box2D

RISC-V market justification

- The Code Speed (performance) is blocking the prosperity of RISC-V
  - Java speed on RV was ~100x slower than x86/Arm w/o JIT
  - V8 and other runtimes w/ JITs are under optimized for RISC-V
  - GCC and Clang still have huge potential to improve
  - So, code speed is important.
- A platform tracking all free available toolchains & runtimes & libs
  - It is important to know where we are, how far can we achieve.
  - It is kind of public goods, lighthouse for toolchains & runtimes & libs
  - So, it is better to put under the RISC-V International

Phasing

- Phase 1: set up the infrastructure
  - one compiler (v8), one board (unleashed), three (JS) benchmarks
  - One engineer, in one month. Run the process. Make the website online.

- Phase 2: more toolchains & runtimes, all available open-source benchmarks.
  - Include upsteram GCC, RISC-V GCC, upstream Clang/LLVM
  - Include OpenJDK/HotSpot, OpenJDK/OpenJ9
  - Include Rust
  - Include Go
  - Include mBench and all available open-source benchmarks

Phasing - cont.

- Phase 3: more physcial boards and libraries
  - Add Hifive Unmatched and other physical boards
  - Set up APIs or processes to let RVI members run the scripts on their own devices and upload to the platform/website.
  - Inlcude OpenBLAS and other compute libs.

Outline plan and timescale

- Phase 1: 1 engineer. 1 month. Plan: 2021-2-1 ~ 2021-3-1
  - Leave one month for volunteers. Can start today, although.
- Phase 2: 2 engineer. 3 month. Plan: 2021-3-1 ~ 2021-6-1
- Phase 3: 2 engineer. 2 month[1]. Plan: 2021-6-1 ~ 2021-8-1
  - [1] it depends on the status of hardware.

Prerequisites

- Basic knowledge about performance testing and tuning
- Basic knowledge about how to establish a website and tuning JS and PHP code.
- Have RISC-V physical board (not necessary if you just want a test)
- Basic knowledge about how to write bash and python scripts.

Deliverables

- An open-source toolset for building compilers & running benchmarks & show perf results on web.
- Data reports generated by this tracking platform.

- A monthly report to the SIG is a routine deliverables for all projects, which will cover
  - key progress in the past month
  - table showing tasks to be done, in progress, complete and verified
  - test/benchmark results
  - updates to the risk register
  - plans for the next month

Milestones

- I think it is just a small project in the first place so the milestone is the phase stage itself.

Costs

- equipment costs
  - Can we just get some boards donated by the member companies made them?
  - An VPS/IaaS for hosting the website. PLCT has existing VPS and can donate for it.
- license costs
  - Use free & open benchmark first.
  - How to donate? Is it possible? (due the LINCENSE of each benchmarks)
- personnel costs
  - TBD. Roughly 2 person 1 day per week?

Risk register

- TBD.

- Risks are assessed by the Impact (I) they have on the project from 1 (minor) to project killer (3) and by the Likelihood (L) of the risk occurring from 1 (10% chance) through 10 (100% chance). The two are multiplied to give an overall Risk Factor (R). Mitigation must be provided for any risk with I = 3 or R >= 10.

Support

- Physical boards are needed.
  - Especially the boards that can run Linux are welcome.
- Members can run scripts in their own boards and upload the data.
- Need commercial toolchains & runtimes run the scripts and send back the performance data to the tracking platform.

- A project is only useful if members are willing to back it with real resources, which can be in kind or in cash. Generally a project should only go ahead under the aegis of RISC-V if multiple members are backing it.
- Table of member organizations and commitments

Welcome to contribute

- This is the first-time proposal.
- It is free to grab. If anyone want to do it, can just rise your hand and bring a detailed proposal next time.
- Need to create a TG? (needs SSC’s approval)