ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

V8 RISCV： Performance evaluation and enhancement first step: find performance baseline

qiuji@iscas.ac.cn

2021/01/07

Outline

01 Goal：Performance evaluation and enhancement of V8-RISCV

02 Evaluation method

03 Counterpart selection：ARM64 CorextA53

04 Current Performance Statistics

05 TODO

1 Goal：performance->how? why? what to do?

the 2021 Wishlist

A few wishes:

- Set up the performance tracking mechanism. (aka. the code speed tracking infrastructure
- Try to support RISC-V Vector Extension (V)
- Try to support B extension (B)
- Try to support P extension (DSP, Andes)
- Try to support C extension (ongoing work by **@derekztu22**)
- Demo tensorflow.js on RISC-V
- Embrace nodejs community and support all the nodejs ecosystem running on RV64GC
- Speed up (blind say 5x) on RV64GC compared with the code we first upstreaming.
- resurrecting more than 10 new interns/contributors/graduates into our project.
- Hope we can see the Chromium running on PicoRio with V8 optimizations (RIOS Lab is working on this).

www

2 Current evaluation method

- Platform:
  - Simulation run: measure count of instructions
  - Native run: measure time (on HiFive Unleashed board)
- benchmark:
  - Synthesis : SunSpider
  - Micro-bench: simple code snippet from https://jsben.ch/browse

3 Counterpart selection

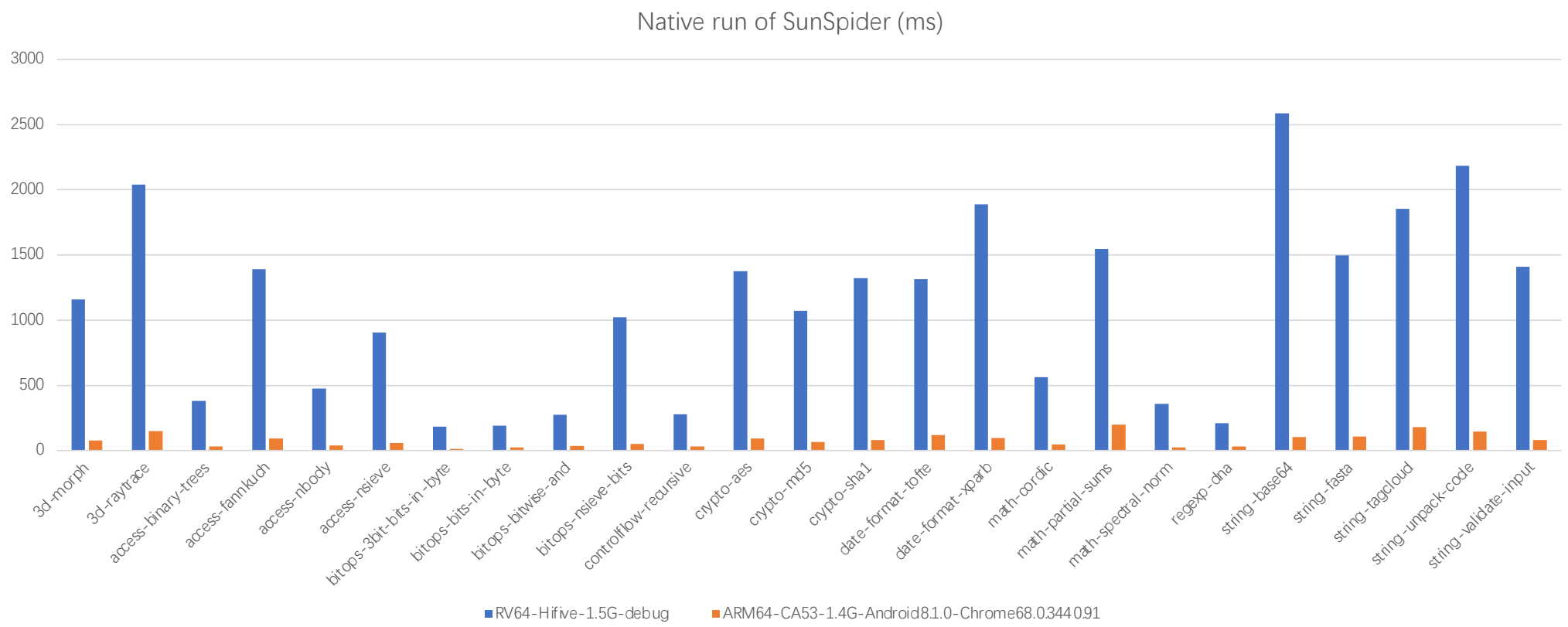
- Final usage scenario: native run embedded in Chrome
  - ARM64 SOCs are more popular than MIPS SOCs

All Standard Cores	Area	Standard Cores	ARM Comparison
E2 Series		E20, E21, E24	M0, M0+, M3, M4, M23, M33
E3 Series		E31 , E34	R4, R5
E7 Series		E76, E76-MC	M7, R7, R8
S2 Series		S21	***
S5 Series		S51, S54	R4, R5
S7 Series		S76, S76-MC	M7, R7, R8
U5 Series		U54, U54-MC	A5, A7, A35, A53
U7 Series		U74, U74-MC	A55
VIU7 Series		VIU75, VIU75-MC	A55
U8 Series		U84	A72, A73

We select A53 because it is the easiest one to get.
Also, ARM64 is a tier1 (or tier2) platform supported by Google.

https://www.sifive.com/cores/u54

4 Performance statistics- SunSpider native run



1. RV64 release build d8 has almost the same result. **(need to checked)**
2. The RV64/ARM64 ratio is from 7x to 20x with an average 14x.

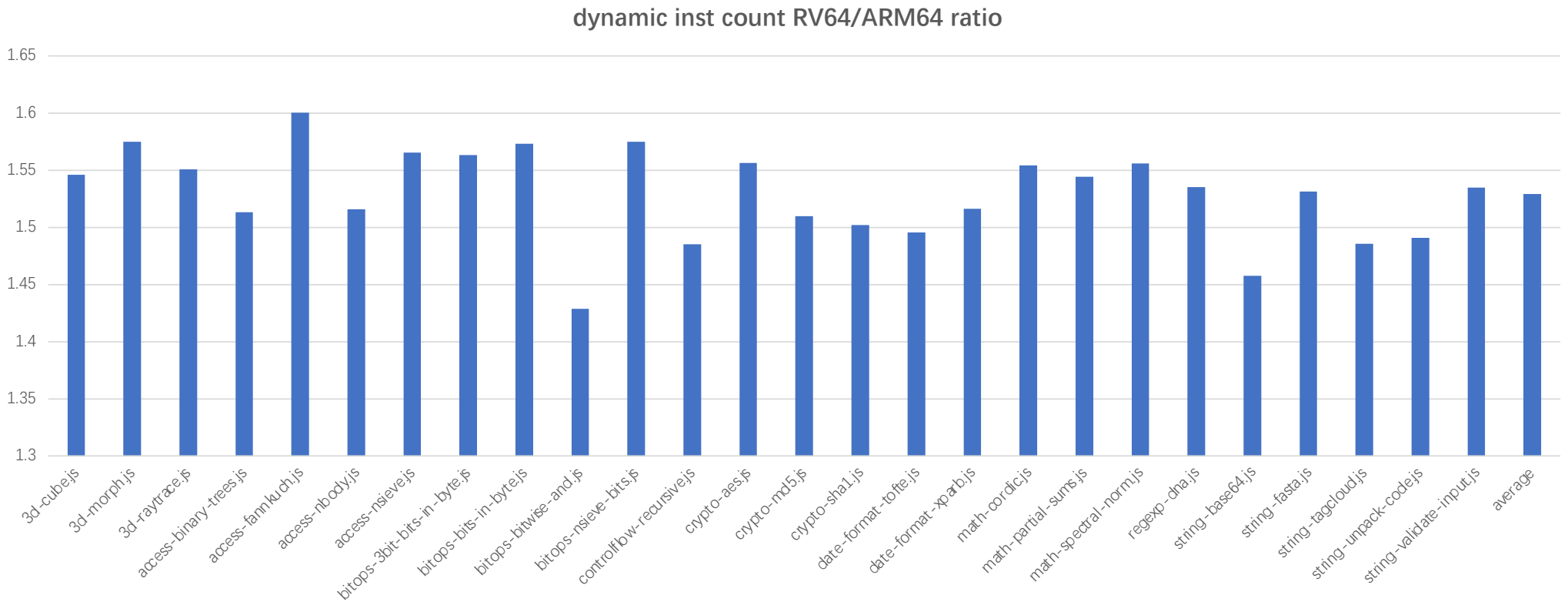
4 Performance statistics- SunSpider simulation run

Dynamic instruction count from the icount_ variable (add the same var to the ARM64 simulator)

	ARM64	RISCV64
3d-cube.js	3.13E+08	4.84E+08
3d-morph.js	1.89E+08	2.97E+08
3d-raytrace.js	2.73E+08	4.23E+08
access-binary-trees.js	8.01E+07	1.21E+08
access-fannkuch.js	2.57E+08	4.11E+08
access-nbody.js	7.47E+07	1.13E+08
access-nsieve.js	1.92E+08	3.01E+08
bitops-3bit-bits-in-byte.js	3.65E+07	5.71E+07
bitops-bits-in-byte.js	3.96E+07	6.24E+07
bitops-bitwise-and.js	6.74E+07	9.63E+07
bitops-nsieve-bits.js	1.91E+08	3.01E+08
controlflow-recursive.js	7.00E+07	1.04E+08
crypto-aes.js	3.16E+10	4.92E+10
crypto-md5.js	1.87E+08	2.82E+08
crypto-sha1.js	2.18E+08	3.28E+08
date-format-tofte.js	1.50E+08	2.24E+08
date-format-xparb.js	3.06E+08	4.65E+08
math-cordic.js	1.03E+08	1.60E+08
math-partial-sums.js	2.05E+08	3.17E+08
math-spectral-norm.js	5.65E+07	8.79E+07
regexp-dna.js	3.75E+08	5.76E+08
string-base64.js	3.48E+08	5.08E+08
string-fasta.js	2.59E+08	3.97E+08
string-tagcloud.js	3.24E+08	4.82E+08
string-unpack-code.js	5.91E+10	8.81E+10
string-validate-input.js	2.30E+11	3.53E+11

4 Performance statistics- SunSpider simulation run

Dynamic instruction count ratio



1. RV64 has executed about 50% more instructions than ARM64.
2. Thera are huge gap between simulation run and native run.**(need to be checked)**

TODO

- make the measurement more accurate, get reasonable result
  - build ARM64 d8 with same gn args and retry
  - profiling on Hifive to see why it takes so long time for the benchmarks
- Try to find optimization potentials
  - find the benchmark which has a biggest performance gap (after solid measurement)
  - segment the binary code into reasonable unit, differentiate and analyze

Q & A

2020/01/07