JS + Numerical Computating
=
Surprising Results!

Vincent Foley-Bourgon

Sable Lab
McGill University

June 2014

Outline

1. Motivation
2. Ostrich
3. Methodology
4. Results
5. Conclusion

Motivation

JavaScript

- Started as utility scripting language (interpreted)
- Now used for industrial-strength applications (advanced JIT compilers)
  - Gmail
  - Google Docs
  - Unreal Engine
- Great for easy deployment
- Available on most modern devices

McLab

- Compiler framework for MATLAB
- Backends for Fortran, X10, .NET and Java
- Does it make sense to target JavaScript?

Is JavaScript suitable for numerical computation?

Sequential

- Is the performance of JavaScript competitive with C?
- Do typed arrays improve the performance of JavaScript code?
- Does the asm.js subset offer performance improvements over hand-written JavaScript?

Parallel

- Does WebCL provide performance improvement versus sequential JavaScript?
- Does WebCL provide performance improvements for JavaScript which are congruent with the performance improvements of OpenCL versus C?

Ostrich

Benchmarking

We need a benchmark suite that:

1. is *representative* of core numerical computations;
2. has *breadth* in the types of applications it represents;
3. has benchmarks that produce *correct results*;
4. allows execution times to be *compared* between languages.

Dwarfs

Dwarf: a group of algorithms with similar properties.

- $\{$ MatMul, LUD $\} \subset$ Dense Linear Algebra
- $\{$ BFS, QuickSort $\} \subset$ Graph Traversal

13 Dwarfs in total that cover the majority of numerical algorithms.

Ostrich

- Covers 12 of the 13 Dwarfs
- Implementations in C, JavaScript, OpenCL and WebCL
- Draws benchmarks from Rodinia and OpenDwarfs
- Open source! `github.com/Sable/Ostrich`

Methodology

Sequential

- C (`gcc -O3`)
- JavaScript w/ typed arrays on Chrome, Firefox, Safari
- JavaScript w/o typed arrays on Chrome, Firefox, Safari
- asm.js on Chrome, Firefox, Safari

Sequential

	Desktop	MacBook Air
CPU	Intel Core i7, 3.20GHz $\times$ 12	Intel Core i7, 1.8GHz $\times$ 2
Cache	12 MiB	4 MiB
Memory	16 GiB	4 GiB
OS	Ubuntu 12.04 LTS	Mac OS X 10.8.5
GCC	4.6.4	llvm-gcc 4.2
Emscripten	1.12.0	1.12.0

Parallel

- C (`gcc -O3`)
- OpenCL
- JavaScript w/ typed arrays on Firefox
- WebCL on Firefox

Parallel

	Tiger	Lion
CPU	Intel Core i7, 2.80GHz $\times$ 8	Intel Core i7, 3.60GHz $\times$ 8
Cache	8 MiB	10 MiB
GPU	NVIDIA Tesla C2050	AMD Radeon HD 7970
Memory	6 GiB	16 GiB
OS	Ubuntu 12.04 LTS	Ubuntu 12.04 LTS
GCC	4.6.3	4.6.3

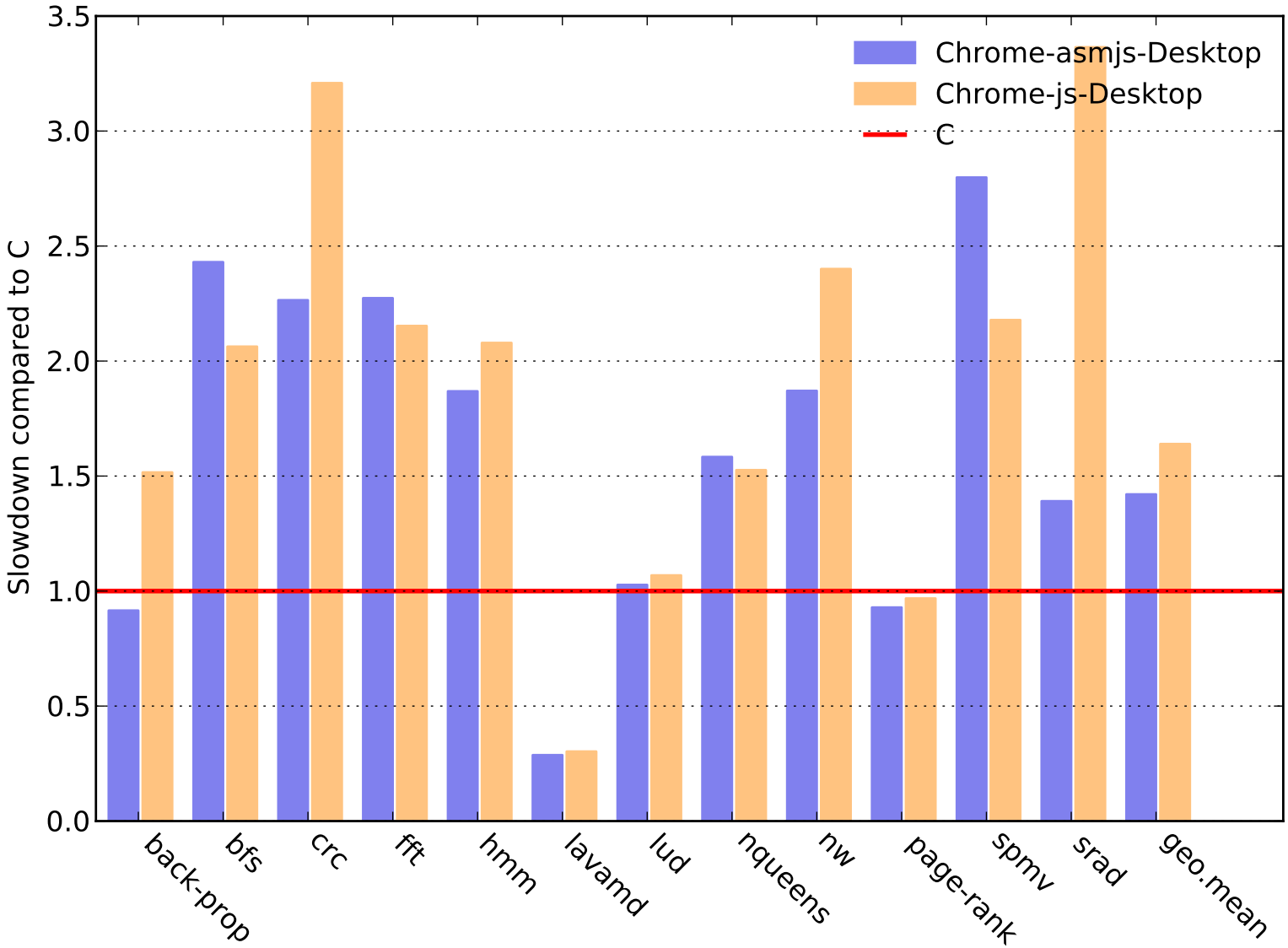
Methodology

- Run benchmarks 10 times
- Time the main algorithm
- Compute the average

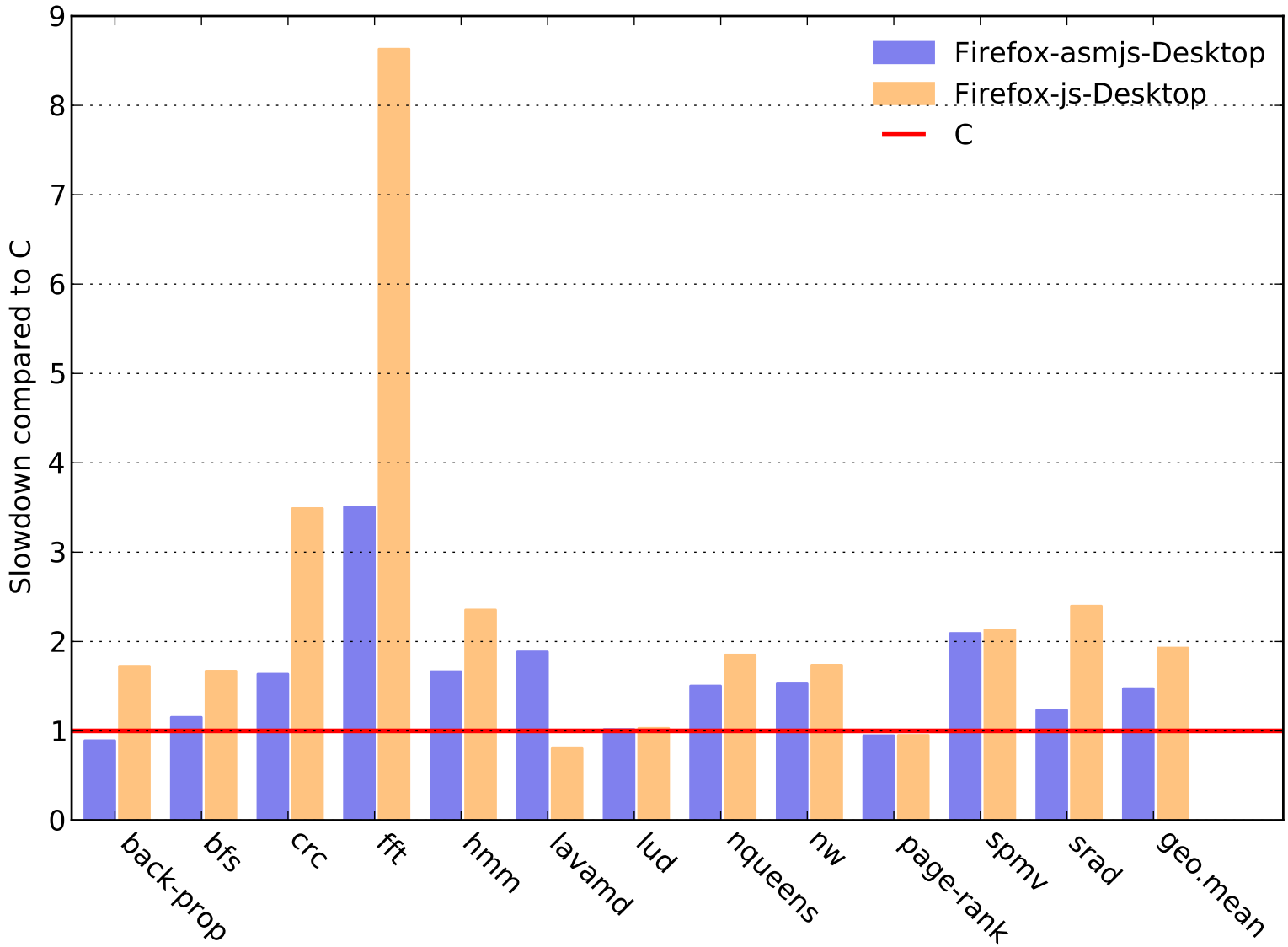
Sequential results

JavaScript vs. C

JS & asm.js vs C, Chrome

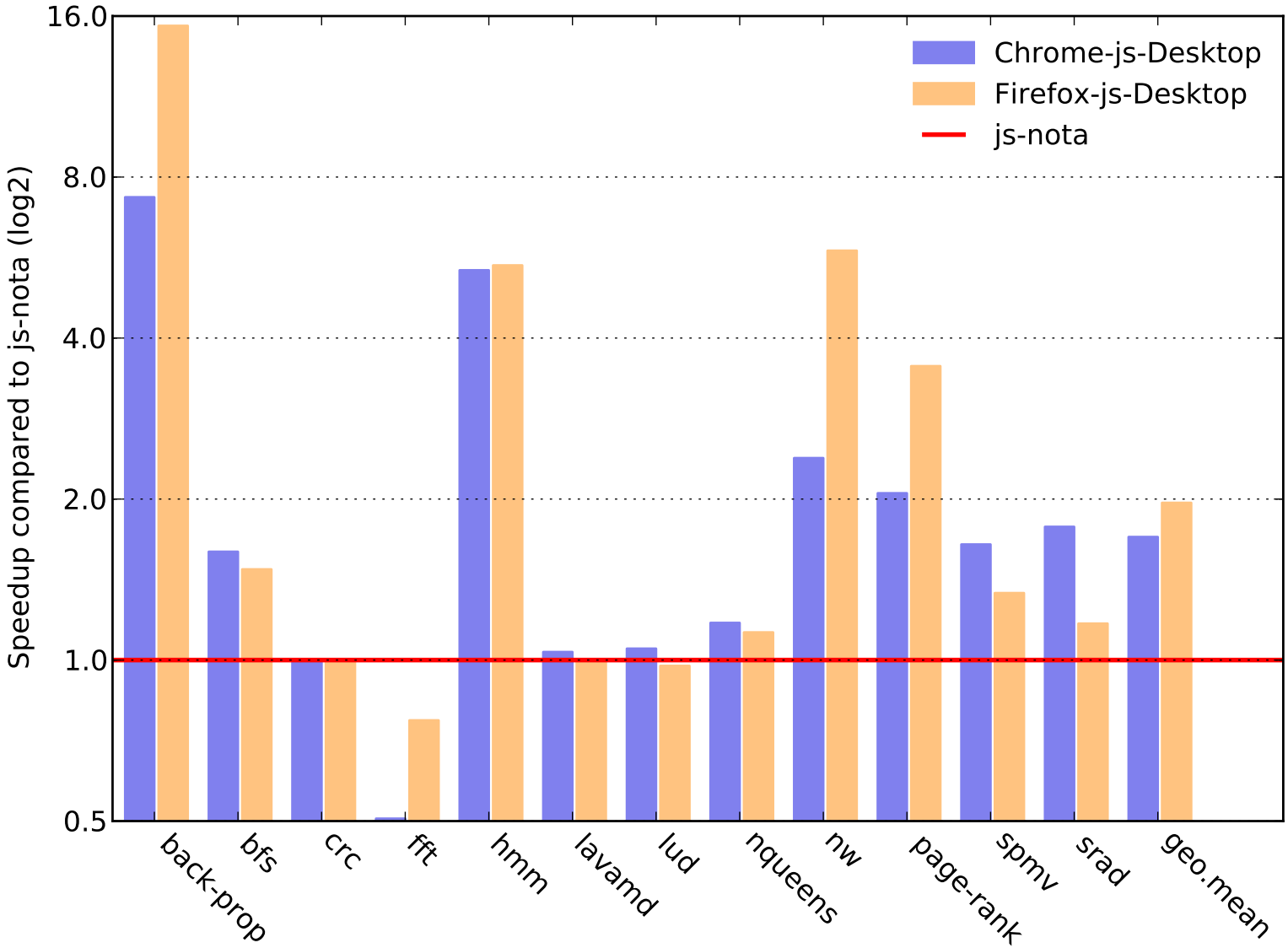


JS & asm.js vs C, Firefox

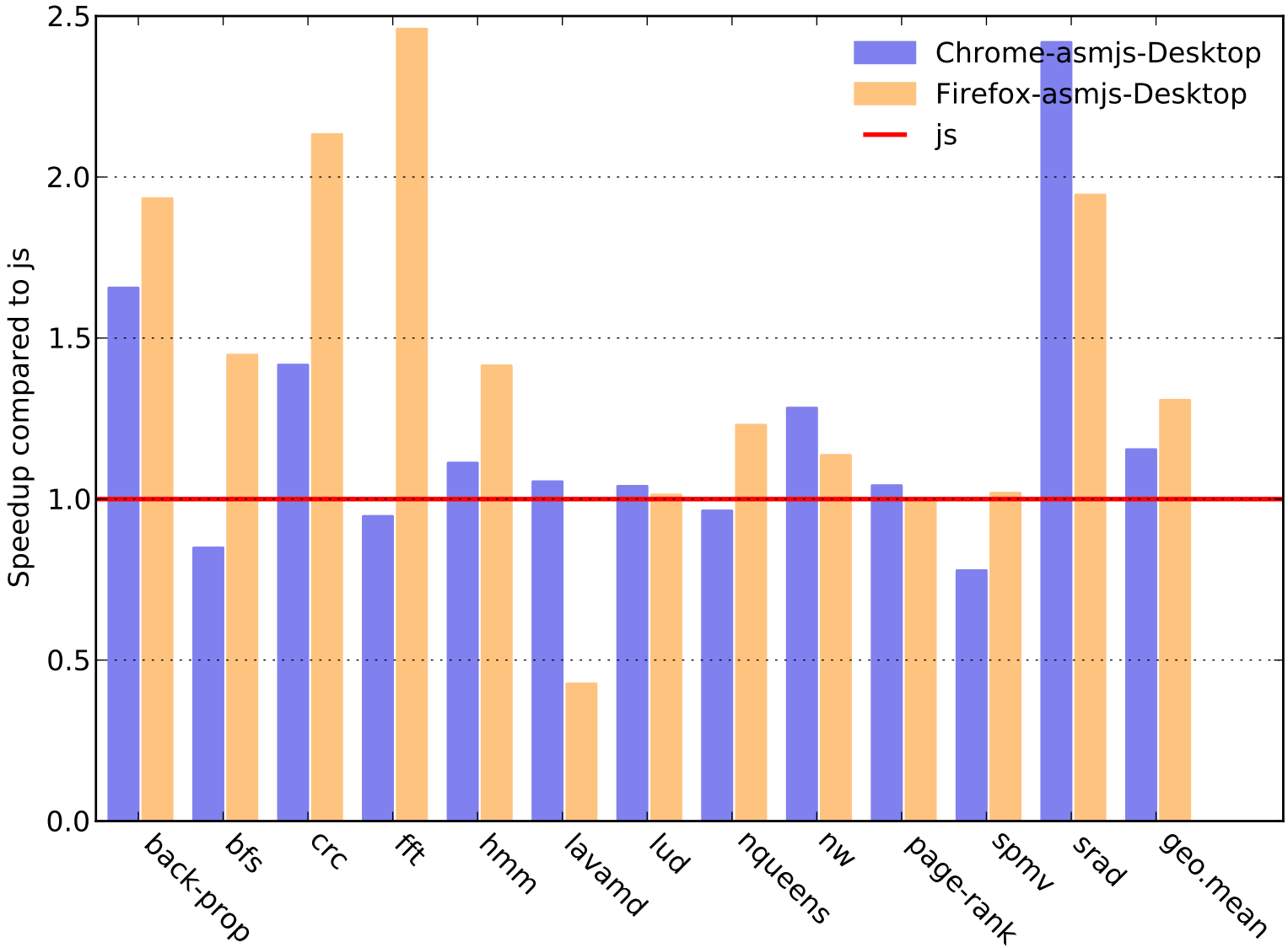


JavaScript vs. JavaScript

JS vs. JS w/o TA

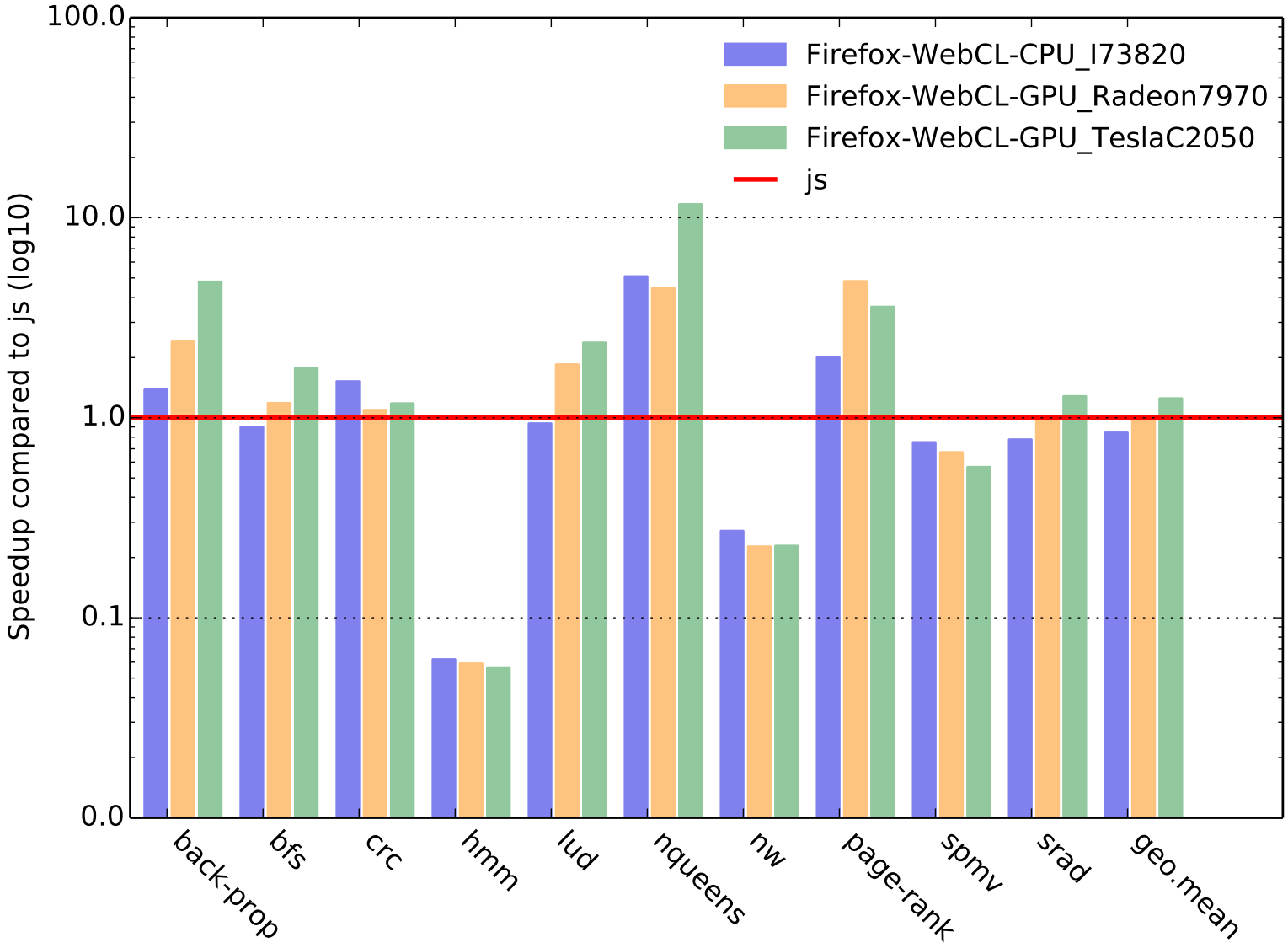

Chrome-js-Desktop
Firefox-js-Desktop
js-nota
Speedup compared to js-nota (log2)
16.0
8.0
4.0
2.0
1.0
0.5
back-prop
bfs
crc
fft
hmm
lavamd
lud
nqueens
nw
page-rank
spmv
srad
geo.mean

asm.js vs. JS

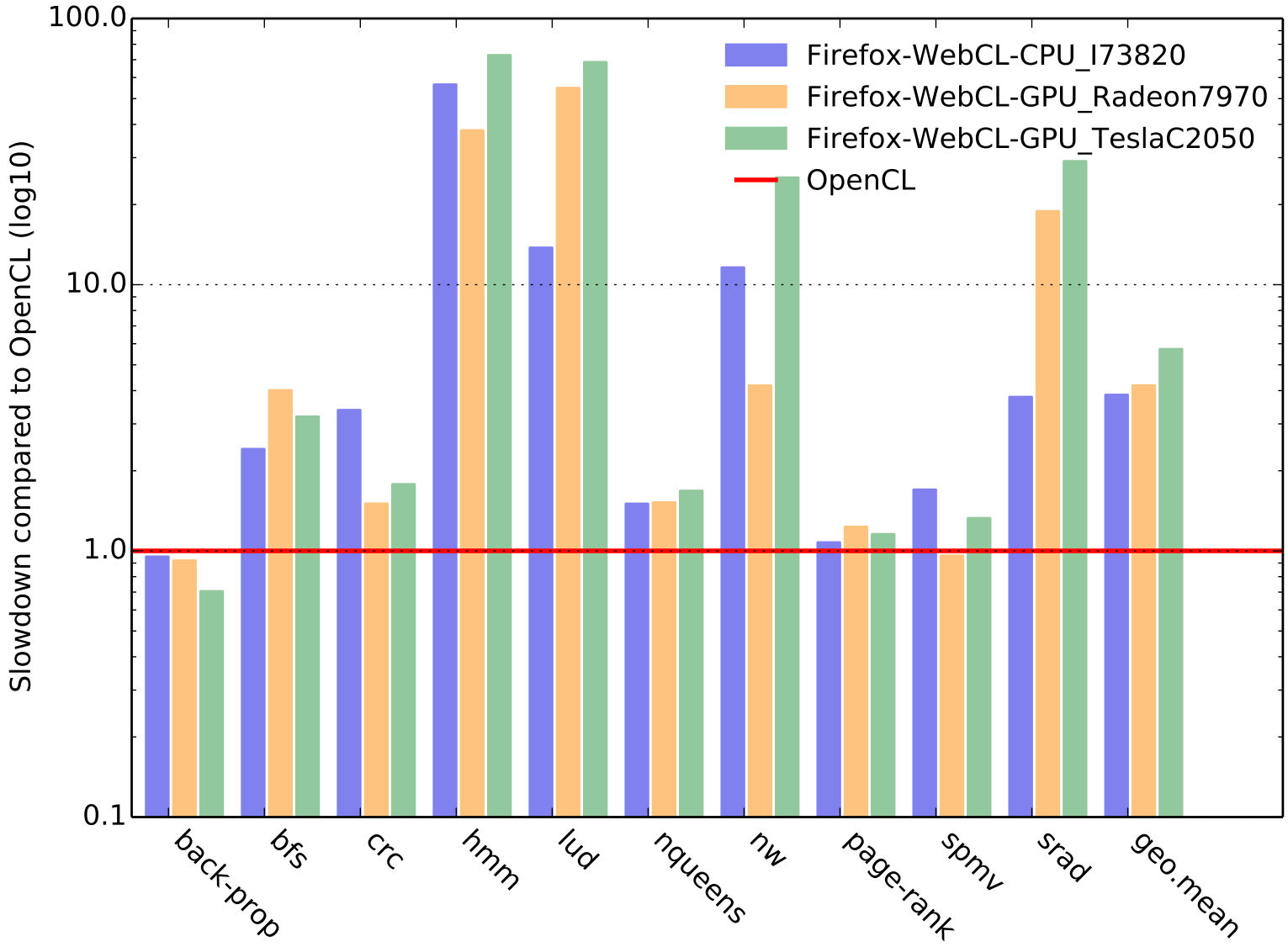

Chrome-asmjs-Desktop
Firefox-asmjs-Desktop
js
Speedup compared to js
2.5
2.0
1.5
1.0
0.5
0.0
back-prop
bfs
crc
fft
hmm
lavamd
lud
nqueens
nw
page-rank
spmv
srad
geo.mean

Parallel results

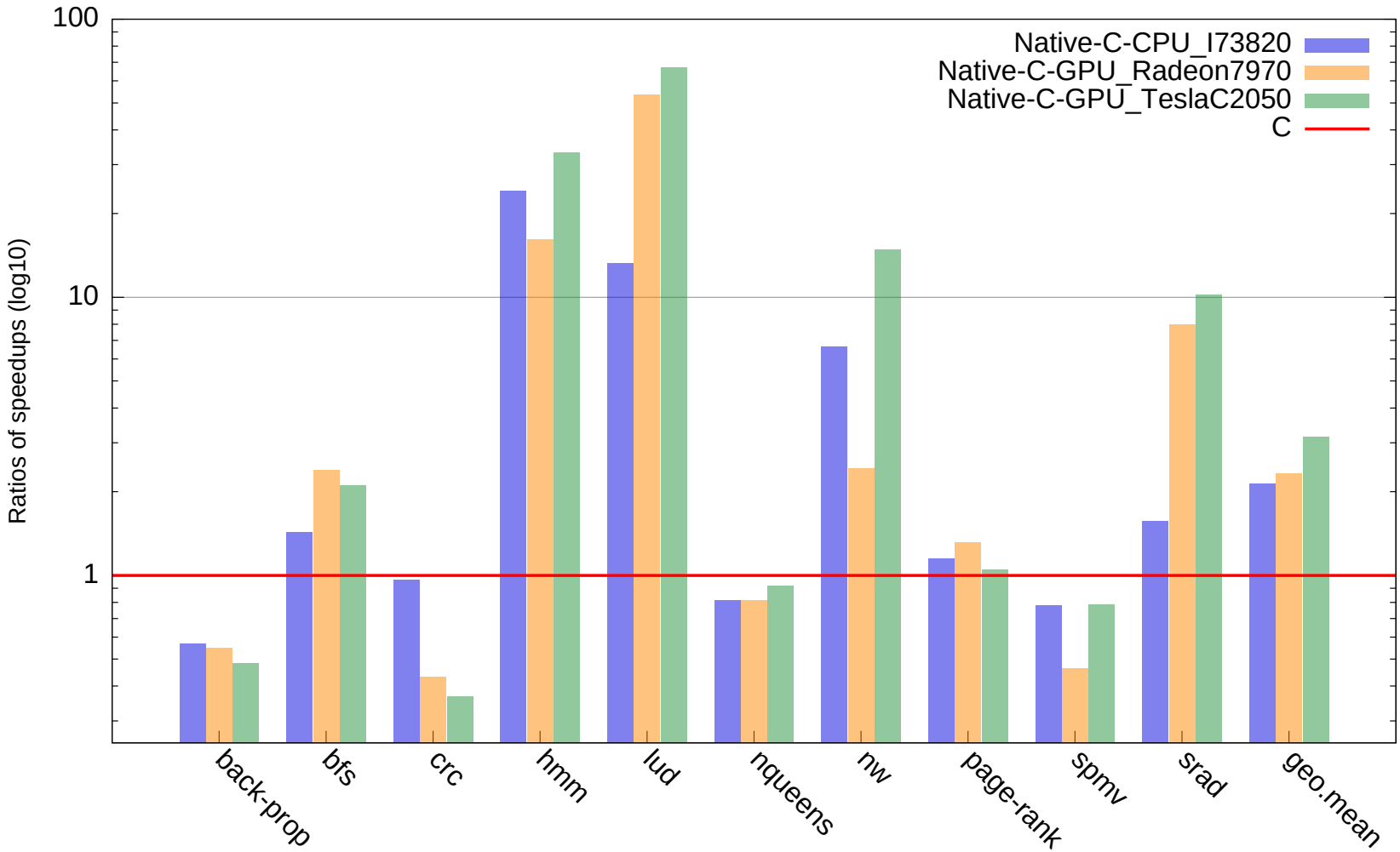
JS vs. WebCL


Firefox-WebCL-CPU_I73820
Firefox-WebCL-GPU_Radeon7970
Firefox-WebCL-GPU_TeslaC2050
js
Speedup compared to js (log10)
100.0
10.0
1.0
0.1
0.0
back-prop
bfs
crc
hmm
lud
nqueens
nw
page-rank
spmv
srad
geo.mean

WebCL vs. OpenCL



WebCL:JS vs. OpenCL:C



Conclusions

Conclusions

Sequential

- In modern browsers, JavaScript can be competitive with C for numerical compuation
- Typed arrays provide a noticeable speedup ($\sim 2x$) in most cases
- asm.js provides a small speedup over JS w/ typed arrays ($\sim 1.2x$)

Conclusions

Parallel

- In benchmarks that can benefit from parallelism, WebCL is advantageous over JavaScript
- The current implementation of WebCL has some overhead costs that make it many times slower than OpenCL

Thank you!

github.com/Sable/Ostrich

www.sable.mcgill.ca/mclab