



JSMeter: Characterizing the Behavior of JavaScript Web Applications

Paruj Ratanaworabhan
Kasetsart University, Thailand

Ben Livshits and Ben Zorn
Microsoft Research, Redmond

in collaboration with

David Simmons, Corneliu Barsan, and Allen Wirfs-Brock



One Word:

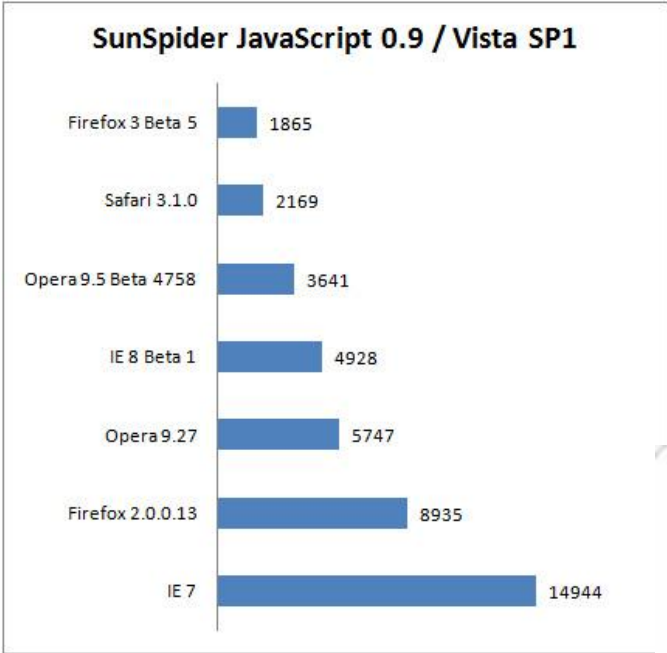
Standard for scripting web applications

Fast JITs widely available

JavaScript

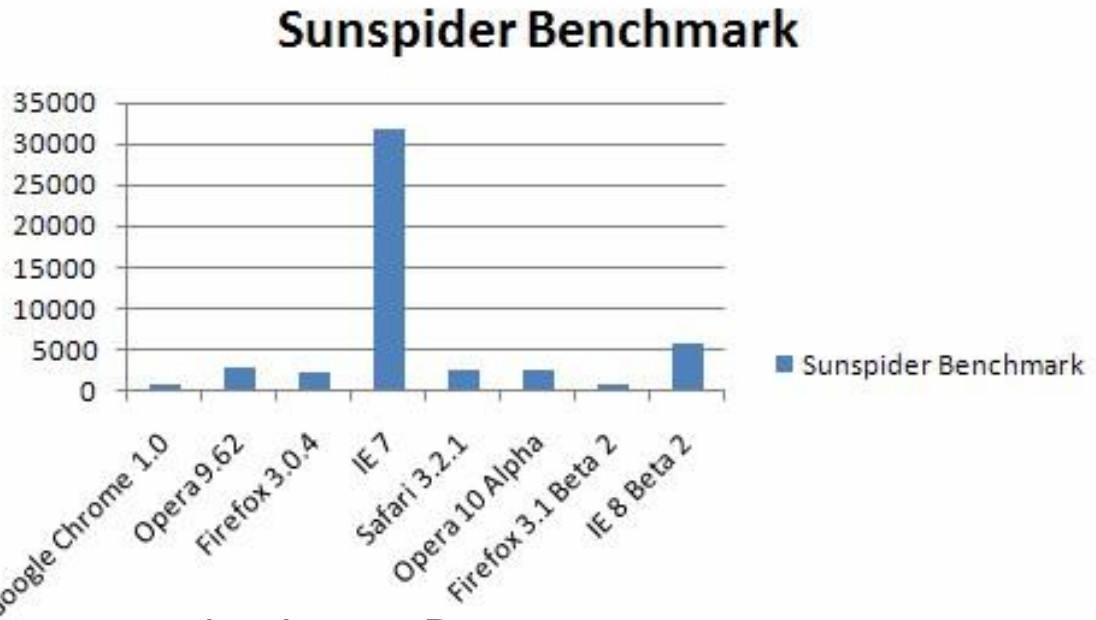
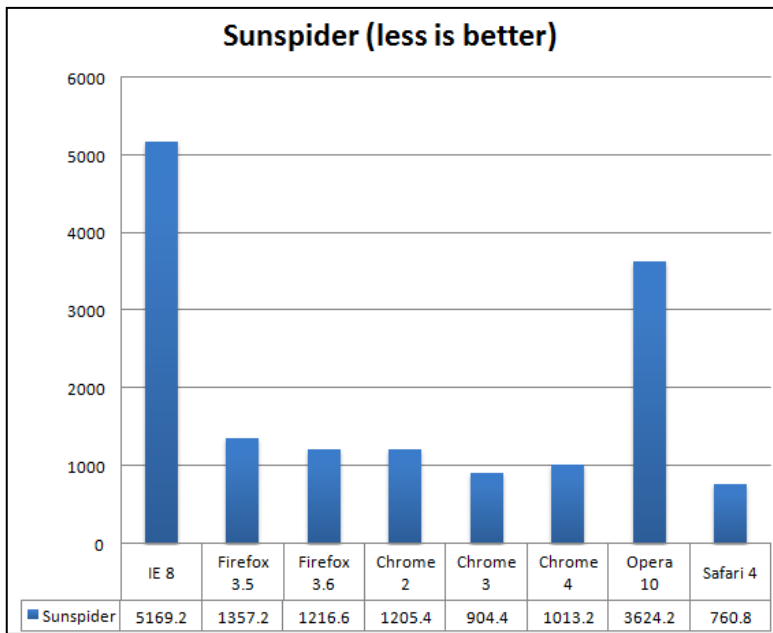
Lots of code present
in all major web sites

Support in every browser



ZDNet 29 May 2008

Browser Wars!



ghacks.net Dec. 2008

Artificial Benchmarks versus Real World Sites

7 V8

programs:

- richards
- deltablue
- crypto
- raytrace

8 SunSpider

programs:

- 3-draytrace
- access-nbody
- bitops-nsieve
- controlflow

JSMeter



11 real sites:



Goals of this Talk

- Show behavior of JavaScript benchmarks versus real sites
- Consider how benchmarks can mislead design decisions

Instrumentation Framework



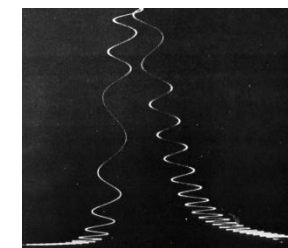
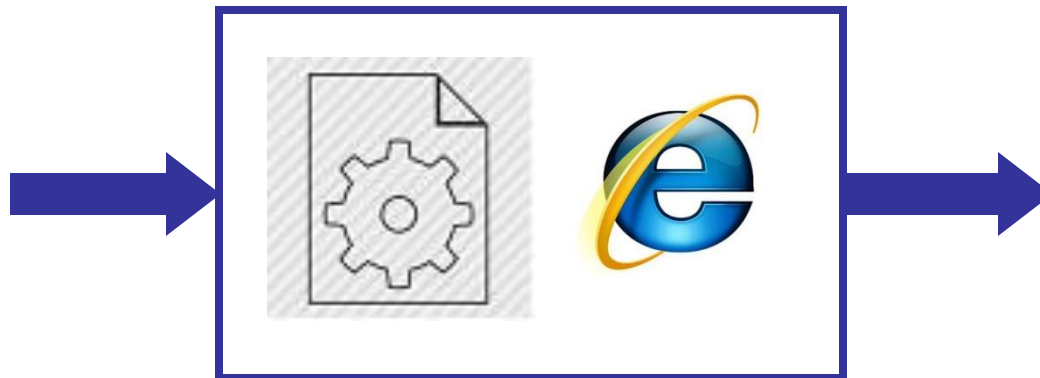
`\ie\jscript*.cpp`



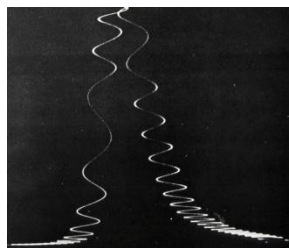
custom jscript.dll



website visits



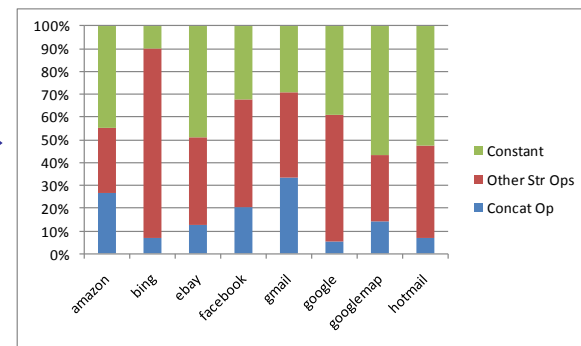
custom trace files



custom trace files



Offline analyzers



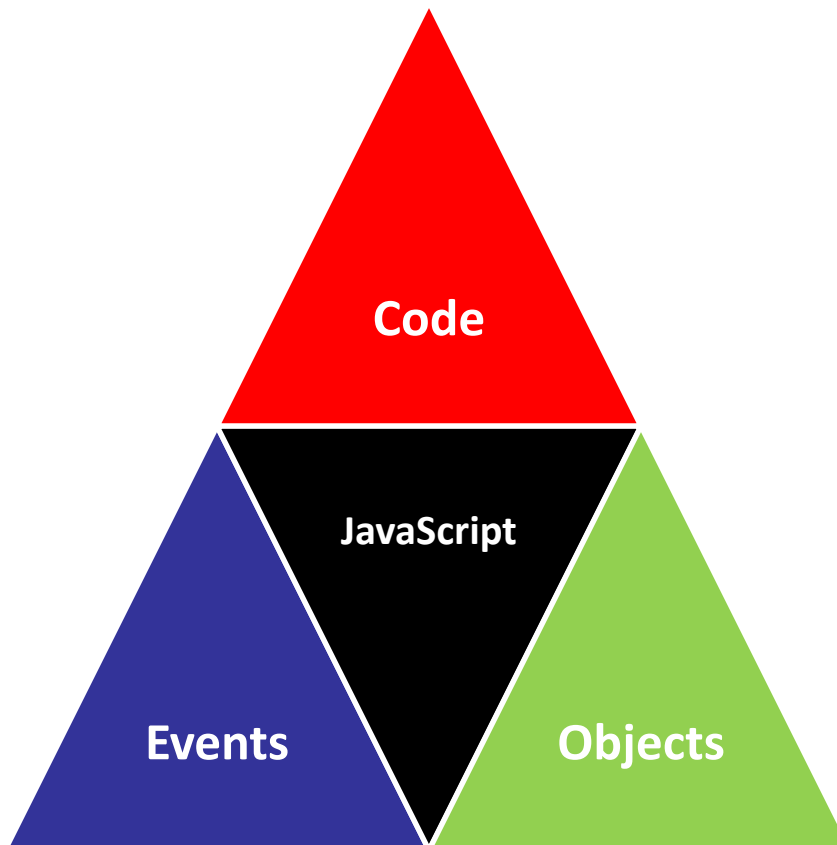
Visiting the Real Sites



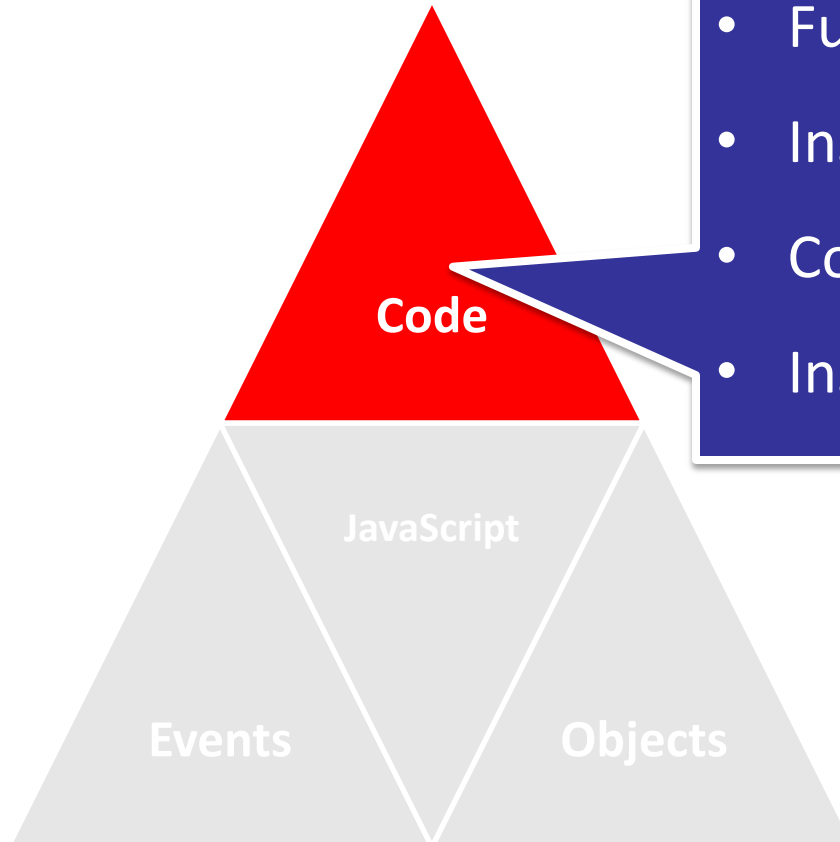
- Getting past page load performance
- Attempted to use each site in “normal” way:

amazon	Search a book, add to shopping cart, sign in, and sign out
bing	Type in a search query and also look for images and news
bingmap	Search for a direction from one city to another
cnn	Read front page news
ebay	Search for a notebook, bid, sign in, and sign out
economist	Read front page news, view comments
facebook	Log in, visit a friend pages, browse through photos and comments
gmail	Sign in, check inbox, delete a mail, and sign out
google	Type in a search query and also look for images and news
googlemap	Search for a direction from one city to another
hotmail	Sign in, check inbox, delete a mail, and sign out

Understanding JavaScript Behavior

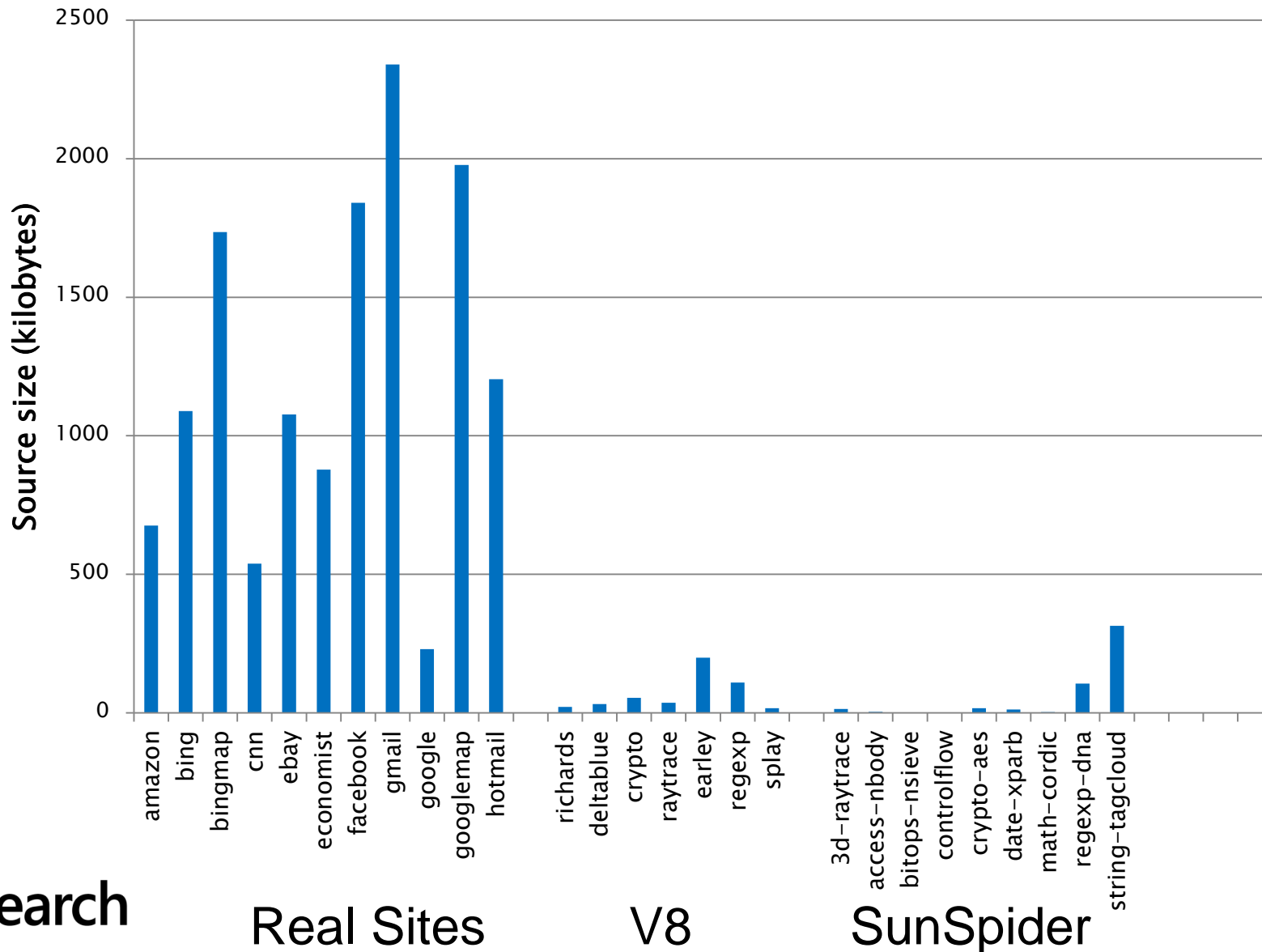


Code Behavior

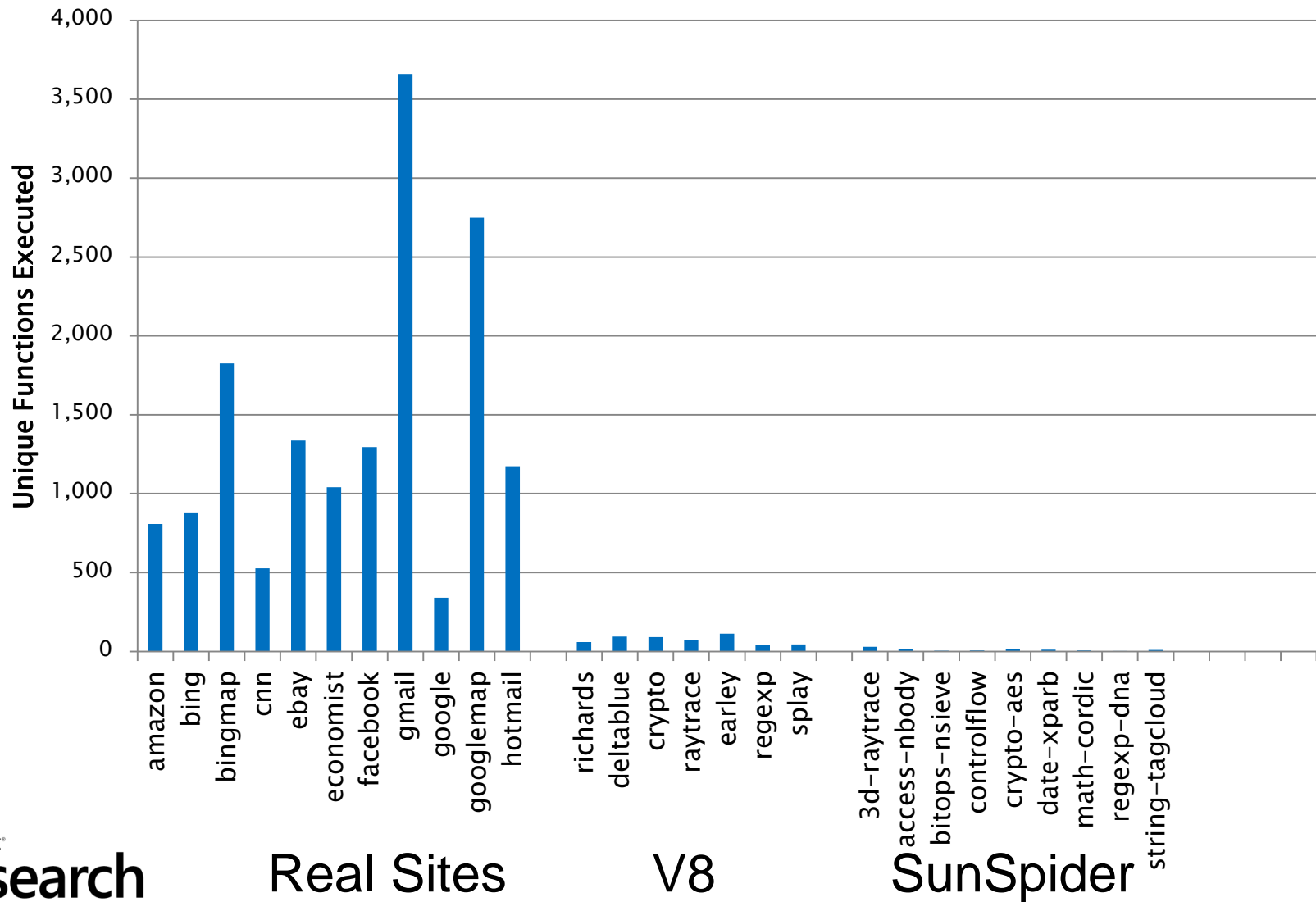


- Function size
- Instructions/call
- Code locality
- Instruction mix

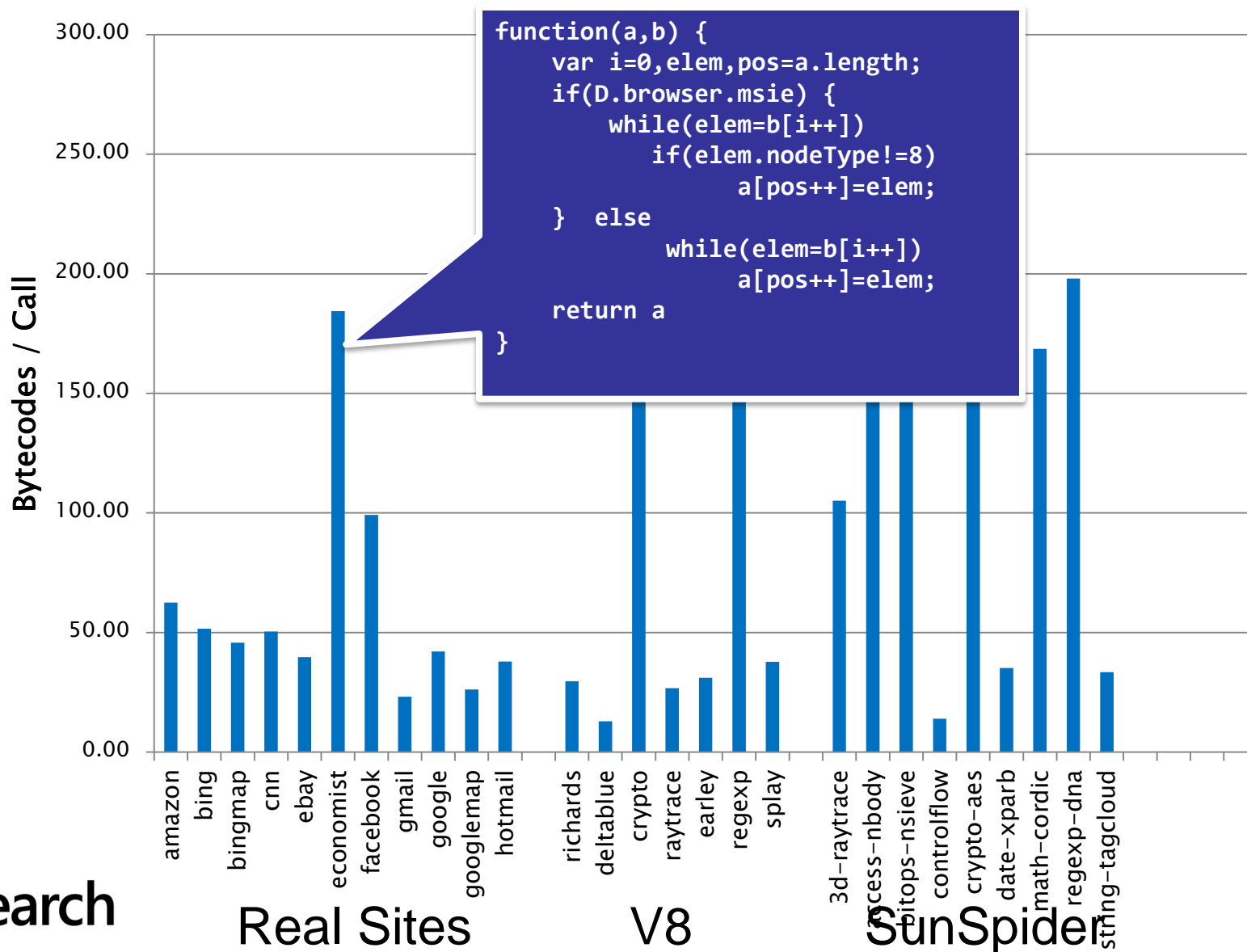
Total Bytes of JavaScript Source



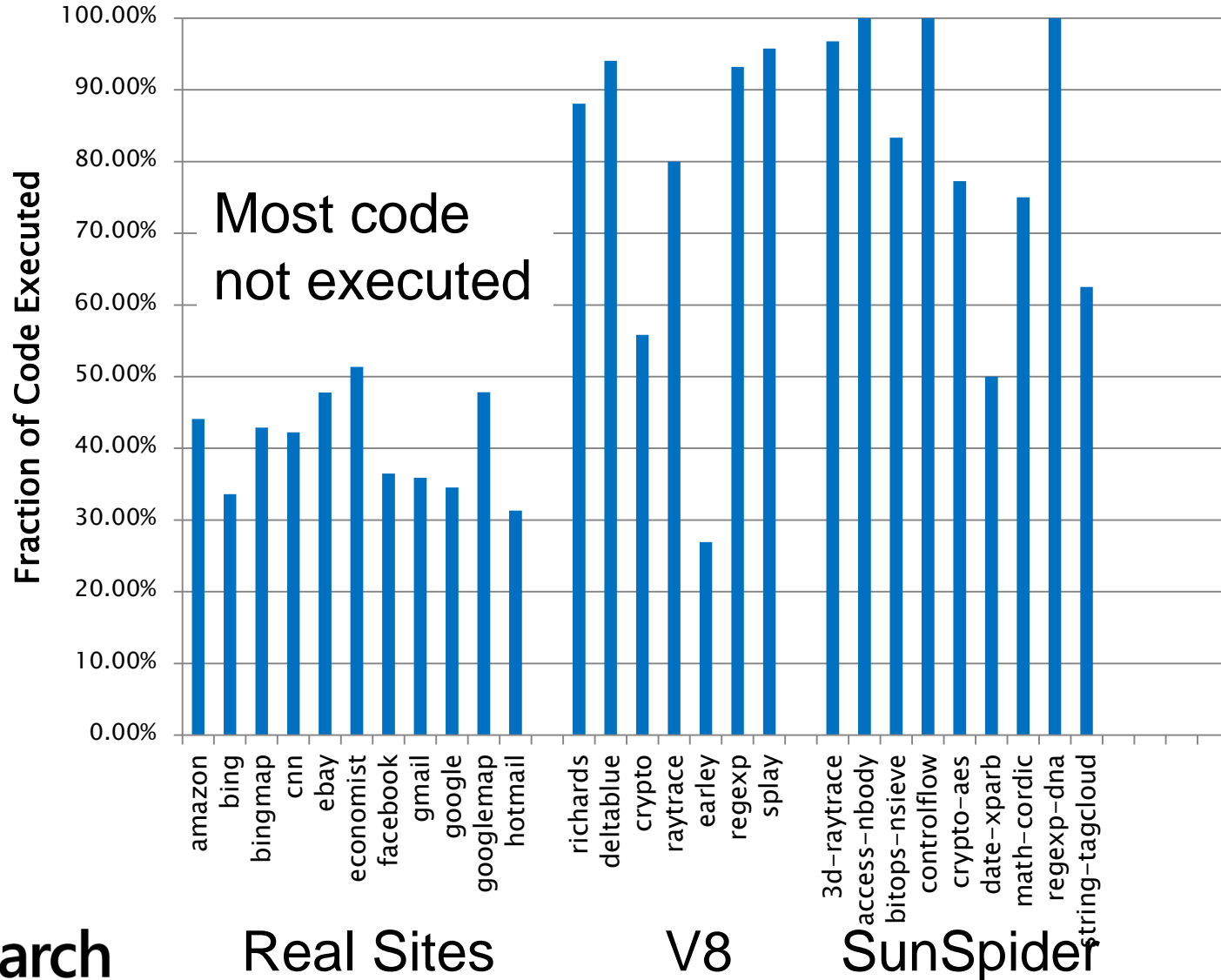
Static Unique Functions Executed



Bytecodes / Call

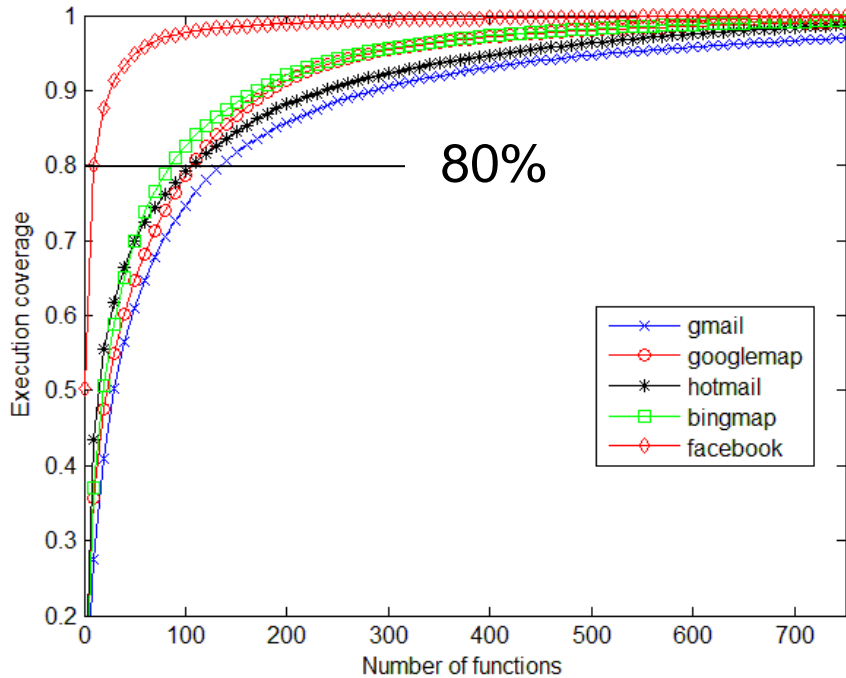


Fraction of Code Executed



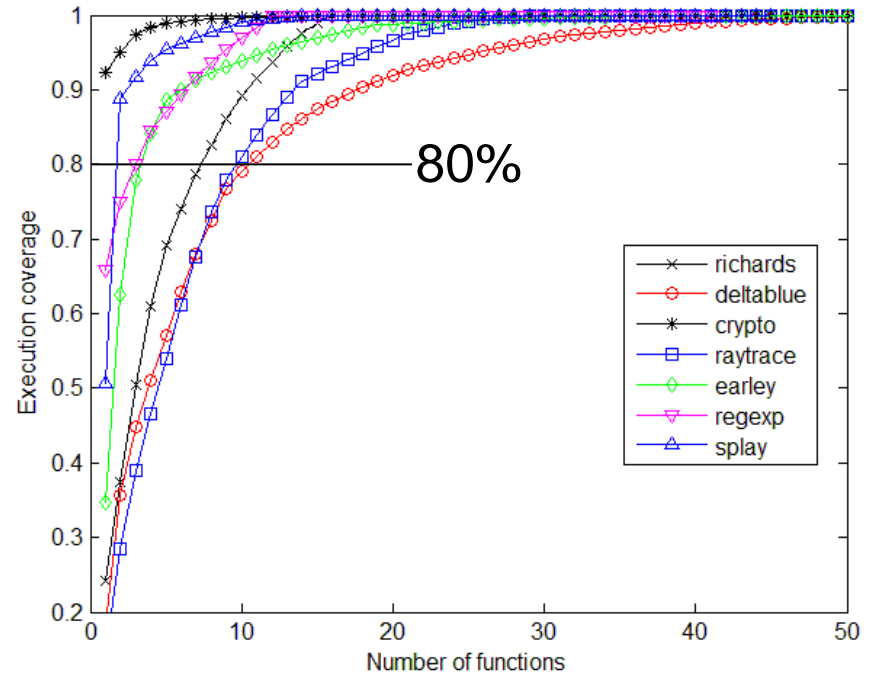
Hot Function Distribution

80% of time in 100+ functions



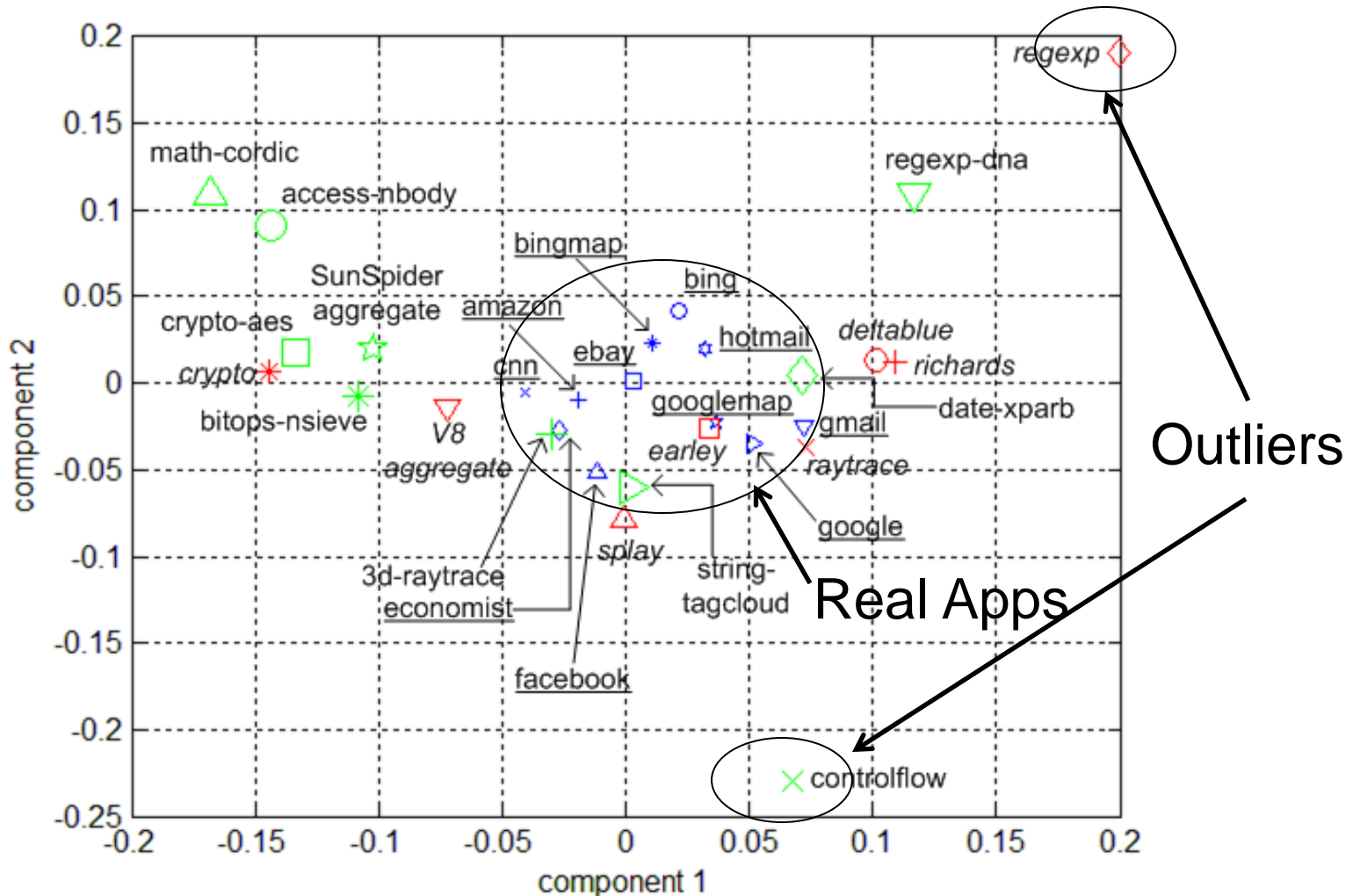
Real Sites

80% of time in < 10 functions



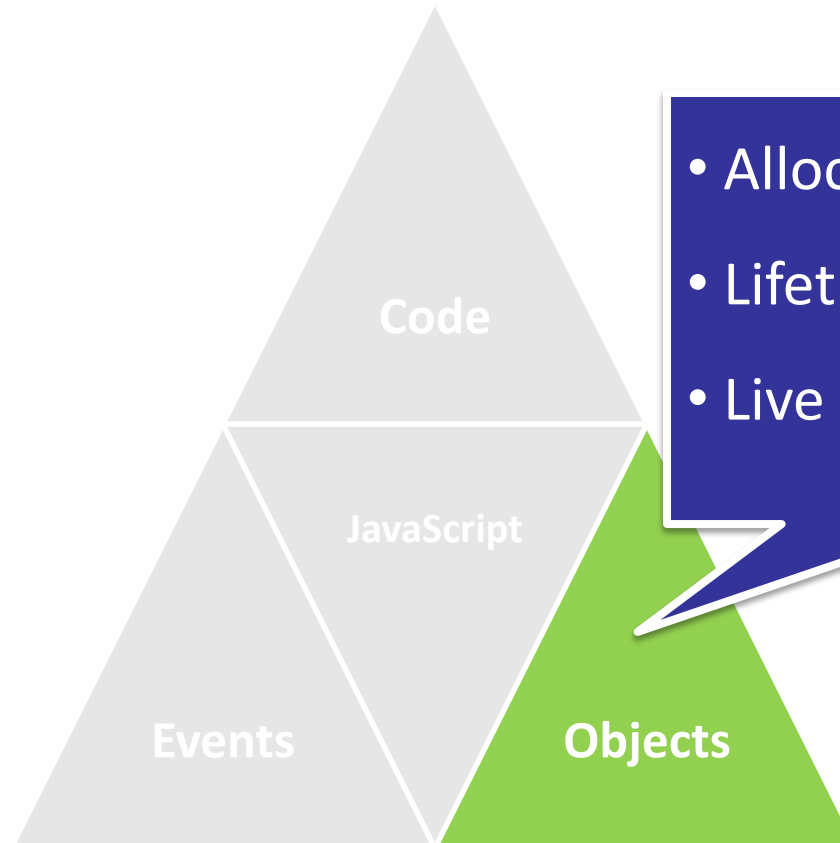
V8 Benchmarks

Opcode Distribution



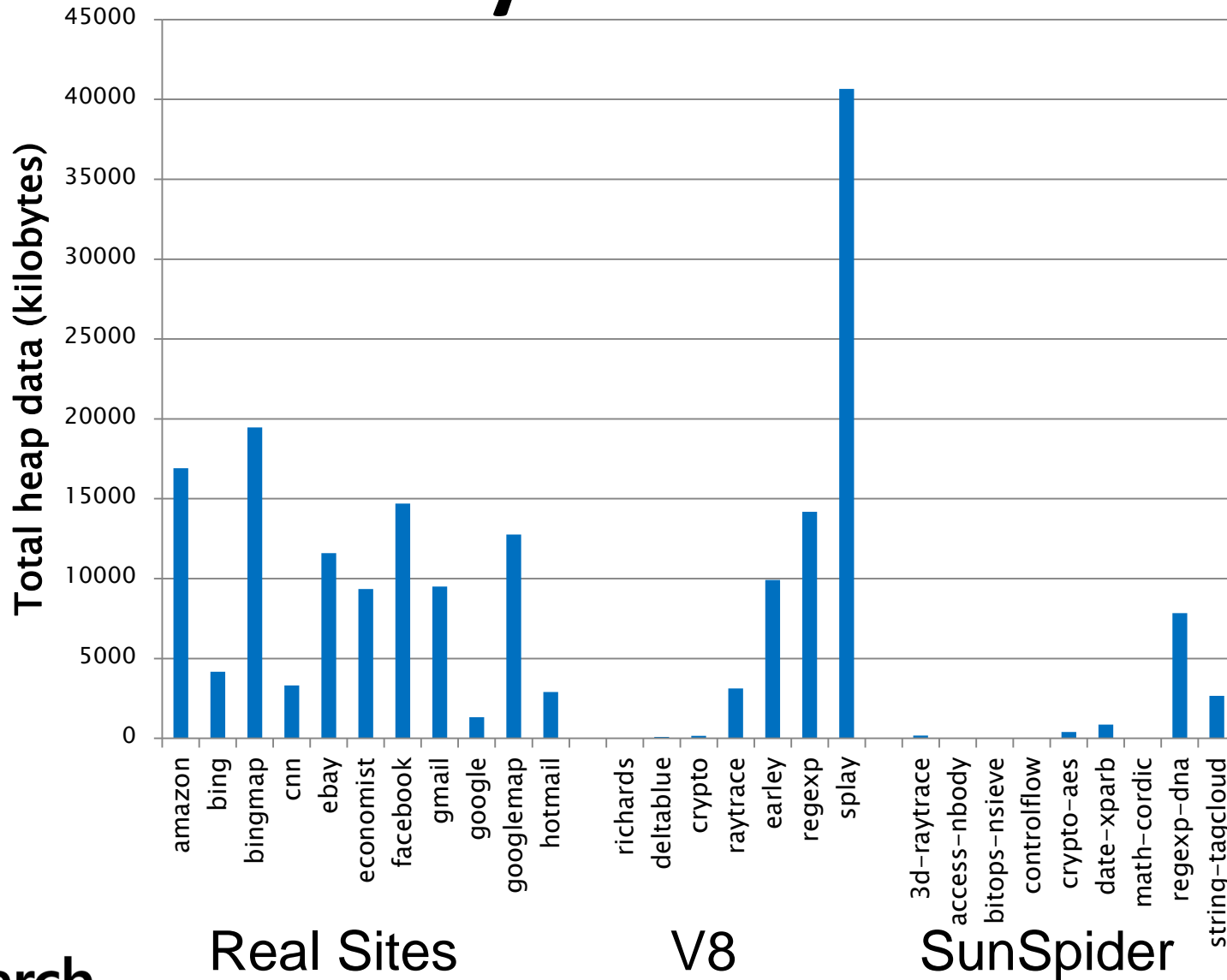
Green = SunSpider | Blue = Real Web Apps | Red = V8

Object Allocation Behavior



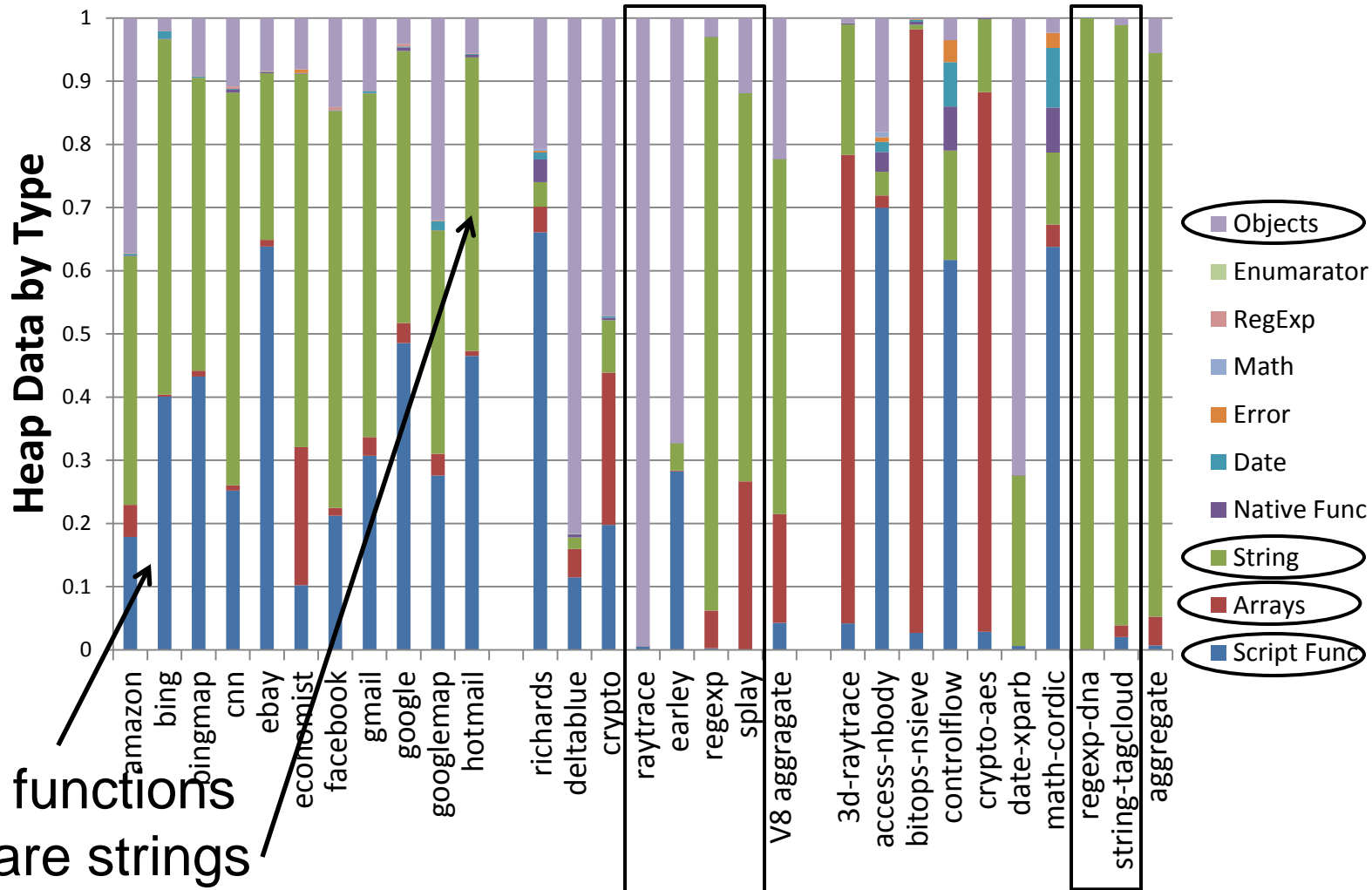
- Allocation by types
- Lifetime distribution
- Live heap composition

Total Bytes Allocated



Heap Data by Type

Few benchmarks allocate much data



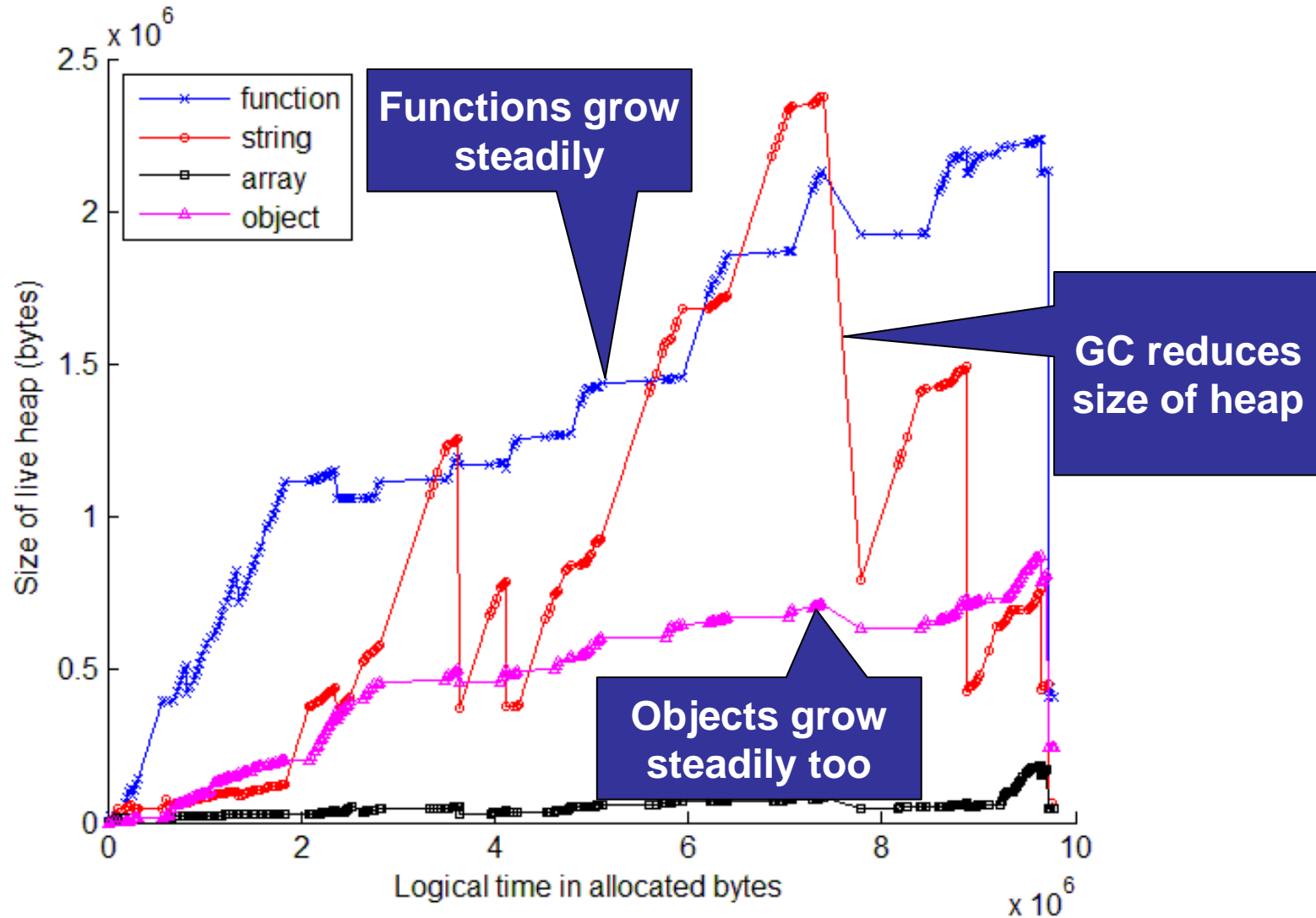
Many functions
Rest are strings

Real Sites

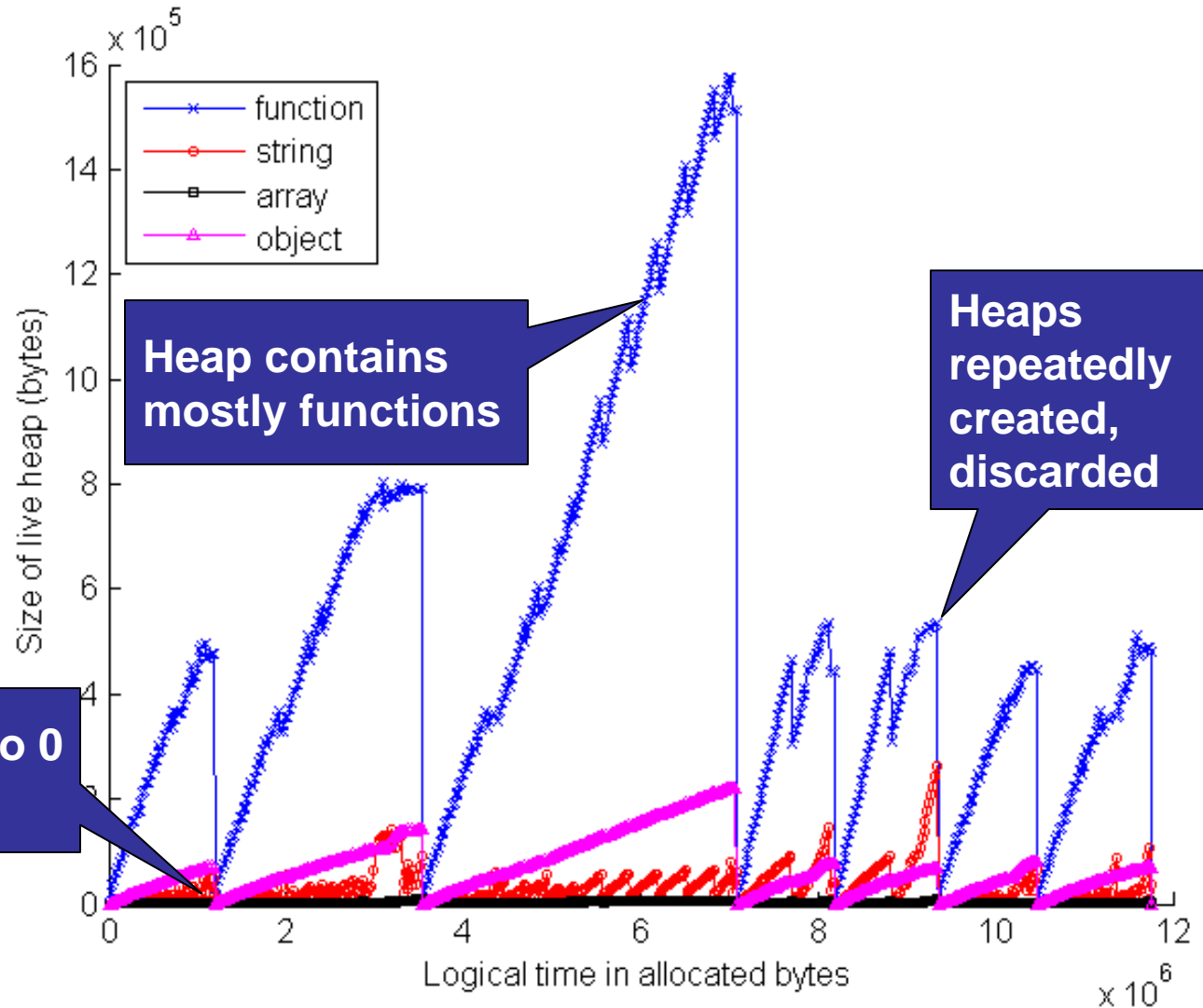
V8

SunSpider

Live Heap Over Time (gmail)

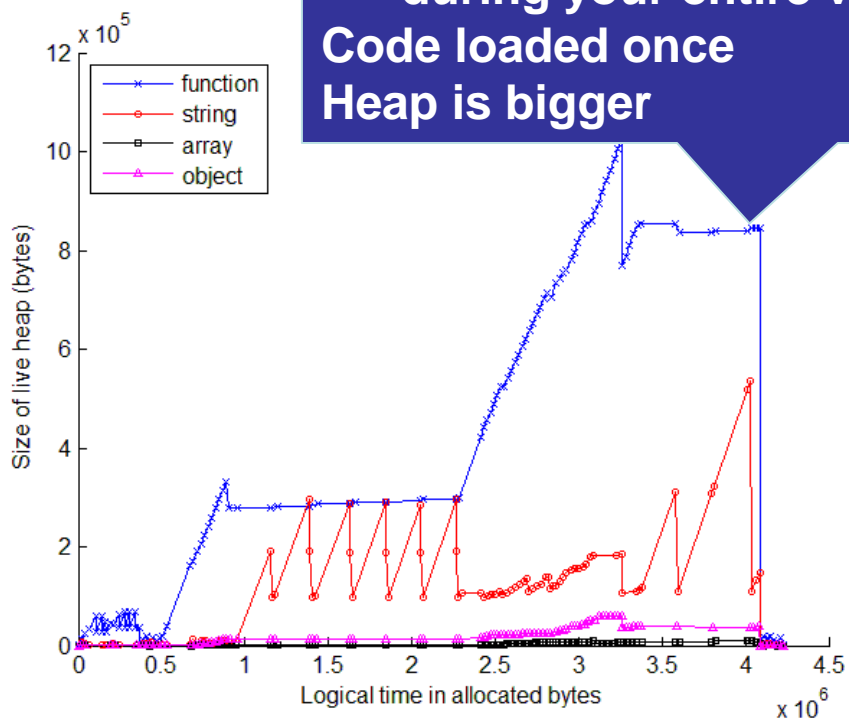


Live Heap over Time (ebay)



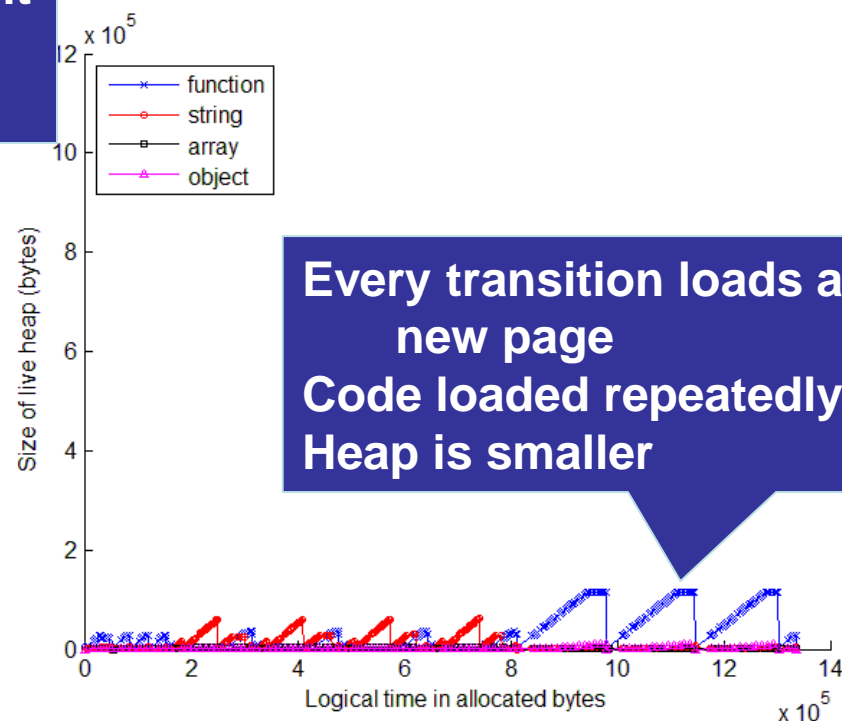
2 Search Websites, 2 Architectures

You stay on the same page during your entire visit
Code loaded once
Heap is bigger



Bing
(Web 2.0)

Every transition loads a new page
Code loaded repeatedly
Heap is smaller

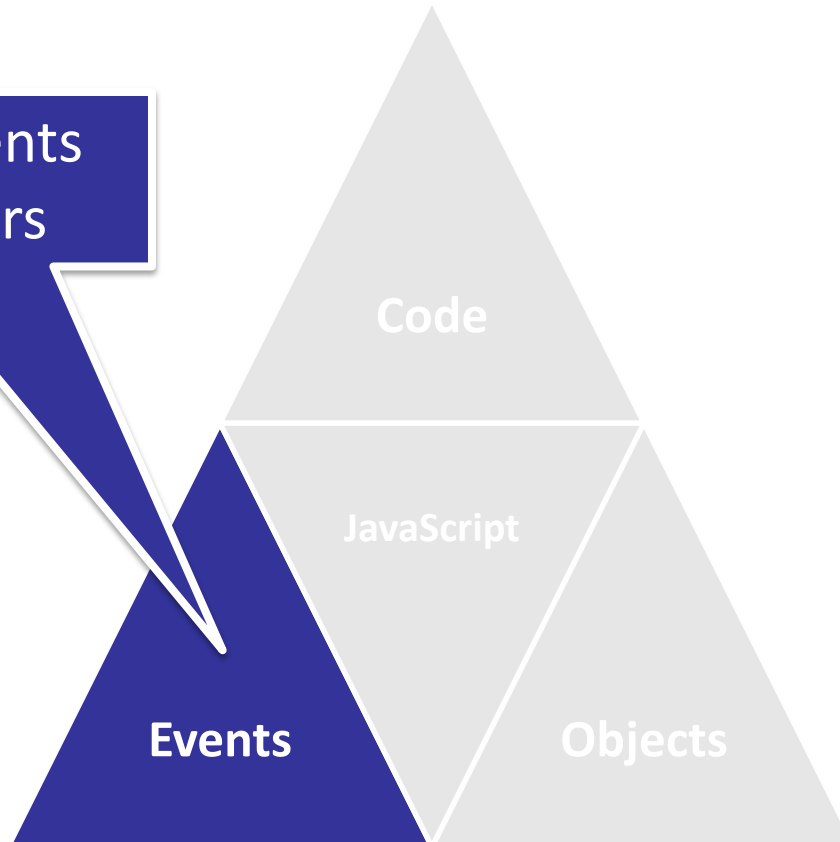


Google
(Web 1.0)

Event Handlers in JavaScript



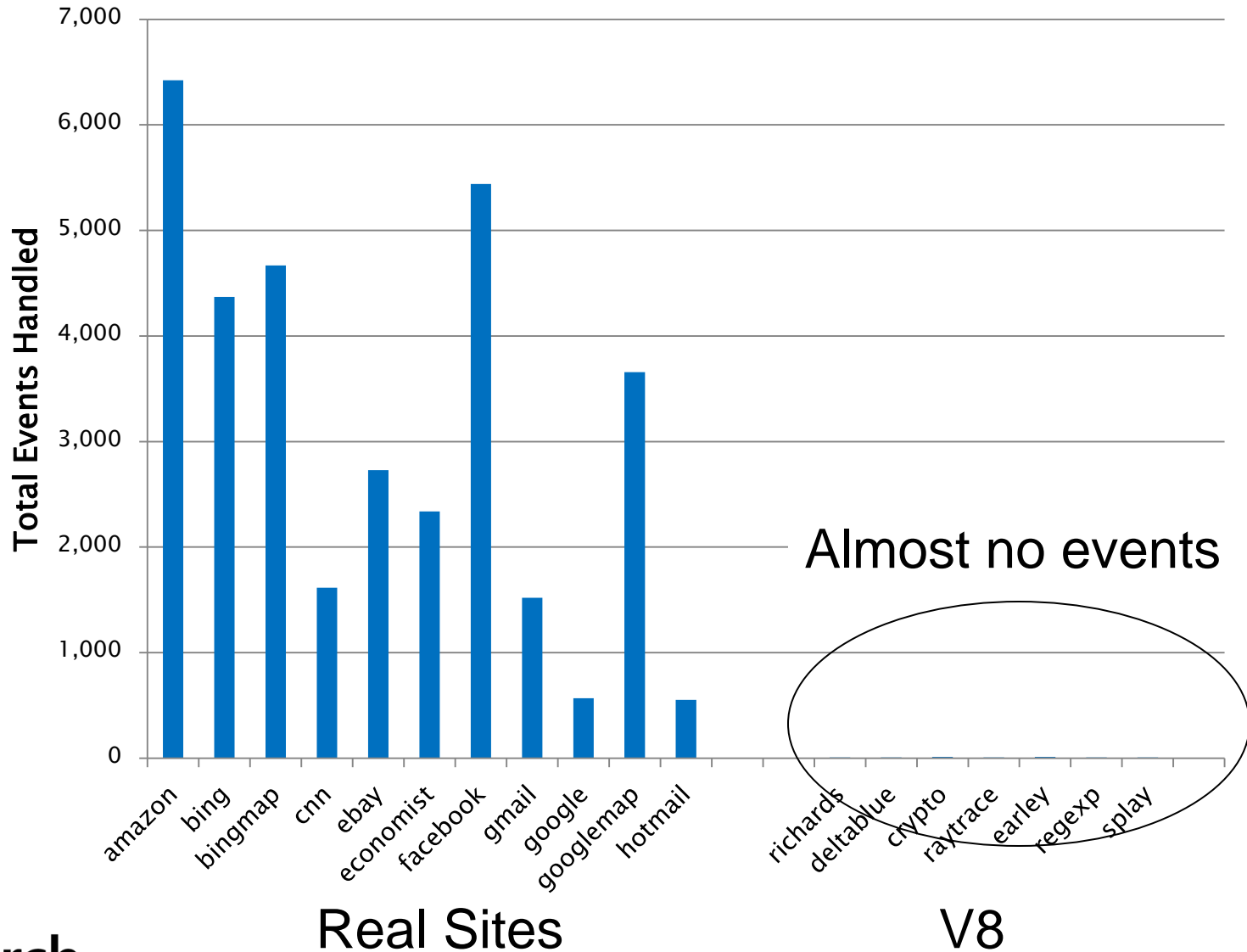
- Number of events
- Sizes of handlers



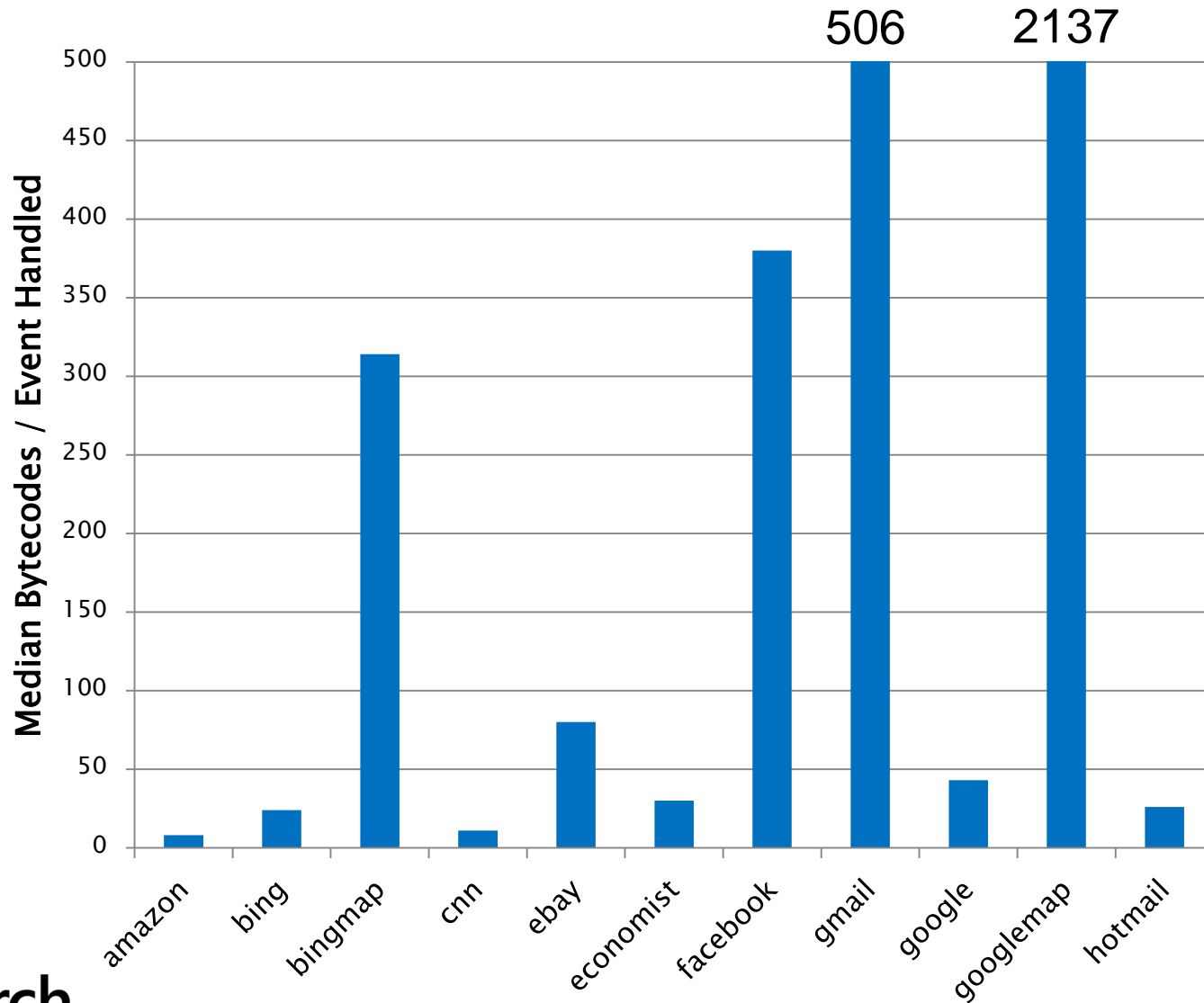
Event-driven Programming Model

- Single-threaded, non-preemptive event handlers
- Example handlers: onabort, onclick, etc.
- Very different from batch processing of benchmarks
- Handler responsiveness critical to user experience

Total Events Handled



Median Bytecodes / Event Handled



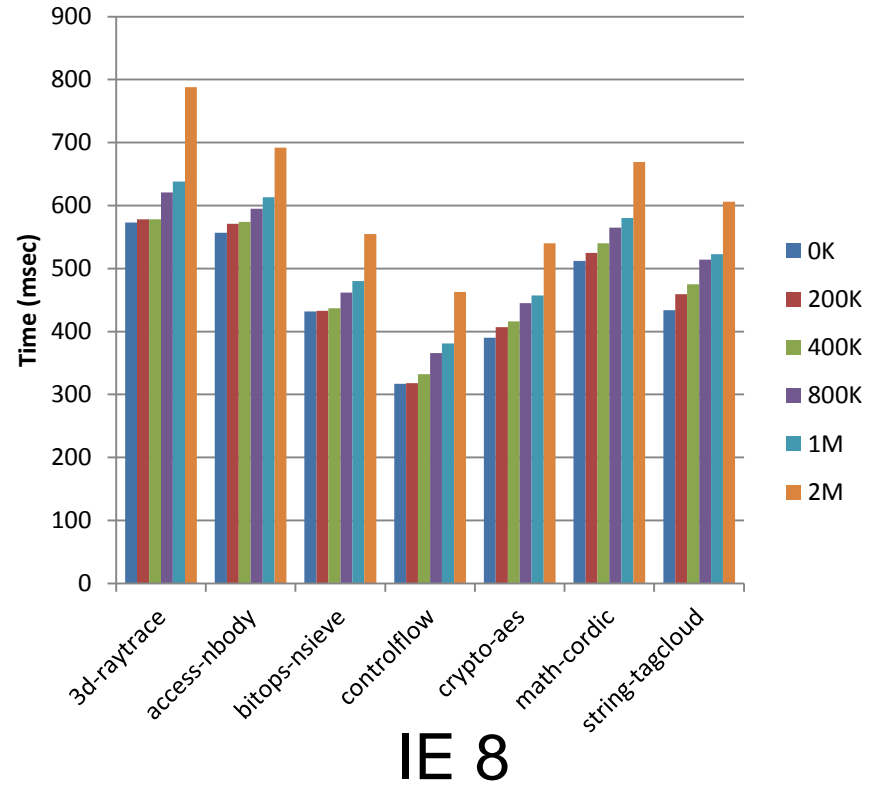
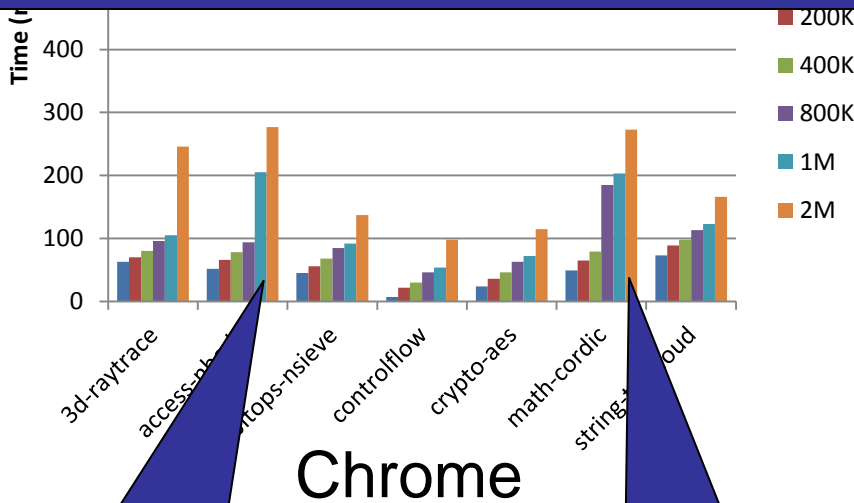
Cold-code Experiment



- Observation
 - Real web apps have lots of code (much of it cold)
 - Benchmarks do not
- Question: What happens if the benchmarks have more code?
 - We added extra, unused to code to 6 SunSpider benchmarks
 - We measured the impact on the benchmark performance

Performance Impact of Cold Code

Without cold code,
 Chrome 12x faster than IE8
 With 2M cold code,
 Chrome 4.7x faster than IE8
 Which comparison is more meaningful?



Cold code makes
 SunSpider on Chrome
 up to 4.5x slower

Cold code has
 non-uniform impact
 on execution time

Impact of Benchmarks



- What gets emphasis
 - Making tight loops fast
 - Optimizing small amounts of code
- Important issues ignored
 - Garbage collection (especially of strings)
 - Managing large amounts of code
 - Optimizing event handling
 - Considering JavaScript context between page loads



Conclusions



- JSmeter is an instrumentation framework
 - Used to measure and compare JavaScript applications
 - High-level views of behavior promote understanding
- Benchmarks differ **significantly** from real sites
 - Misleads designers, skews implementations
- Next steps
 - Develop and promote better benchmarks
 - Design and evaluate better JavaScript runtimes
 - Promote better performance tools for JavaScript developers

Conclusions



- JSMeter is an instrumentation framework
 - Used to measure and compare JavaScript applications
 - High-level views of behavior promote understanding
- Behavior in benchmarks **significantly** differs from real sites
 - Misleads designers, skews implementations
- Next steps
 - Develop and promote better benchmarks
 - Design and evaluate better JavaScript runtimes
 - Promote better performance tools for JavaScript developers

Additional Resources



- **Project:** <http://research.microsoft.com/en-us/projects/jsmeter/>
- **Video:** [Project JSMeter: JavaScript Performance Analysis in the Real World](#)" - MSDN Channel 9 interview with Erik Meier, Ben Livshits, and Ben Zorn
- **Papers:**
 - “JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications”, Paruj Ratanaworabhan, Benjamin Livshits and Benjamin G. Zorn, USENIX 2010 Conference on Web Application Development (WebApps’10), June 2010.
 - ["JSMeter: Characterizing Real-World Behavior of JavaScript Programs"](#), Paruj Ratanaworabhan, Benjamin Livshits, David Simmons, and Benjamin Zorn, MSR-TR-2009-173, December 2009 (49 pages), November 2009.

Additional Slides

Related Work

- JavaScript
 - “An Analysis of the Dynamic Behavior of JavaScript Programs”, Gregor Richards, Sylvain Lebesne, Brian Burg, Jan Vitek, PLDI 2010, June 2010/
- C# and Java
 - Doufour
- C++