RiSE

RiSE Research in Software Engineering

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

Microsoft Research

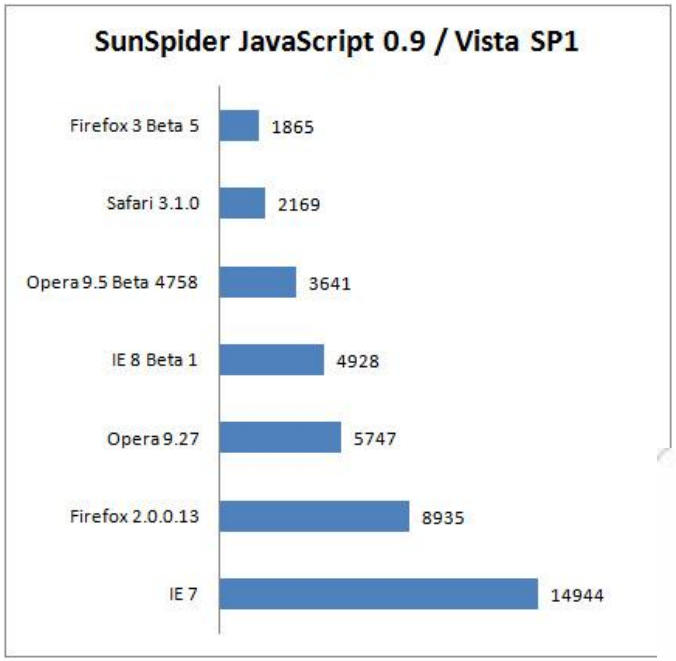
RiSE

Why Measure JavaScript?

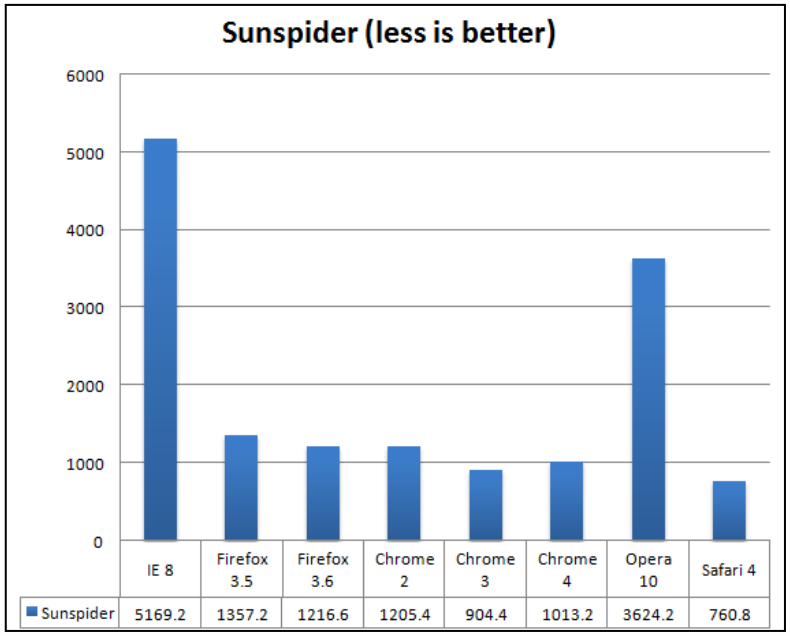
- Standardized, de facto language for the web
  - Support in every browser, much existing code
- Browser and JavaScript performance is important
  - Are current JavaScript benchmarks representative?
  - Limited understanding of JavaScript behavior in real sites
- Who cares?
  - Users, web application developers, JavaScript engine developers

Microsoft Research

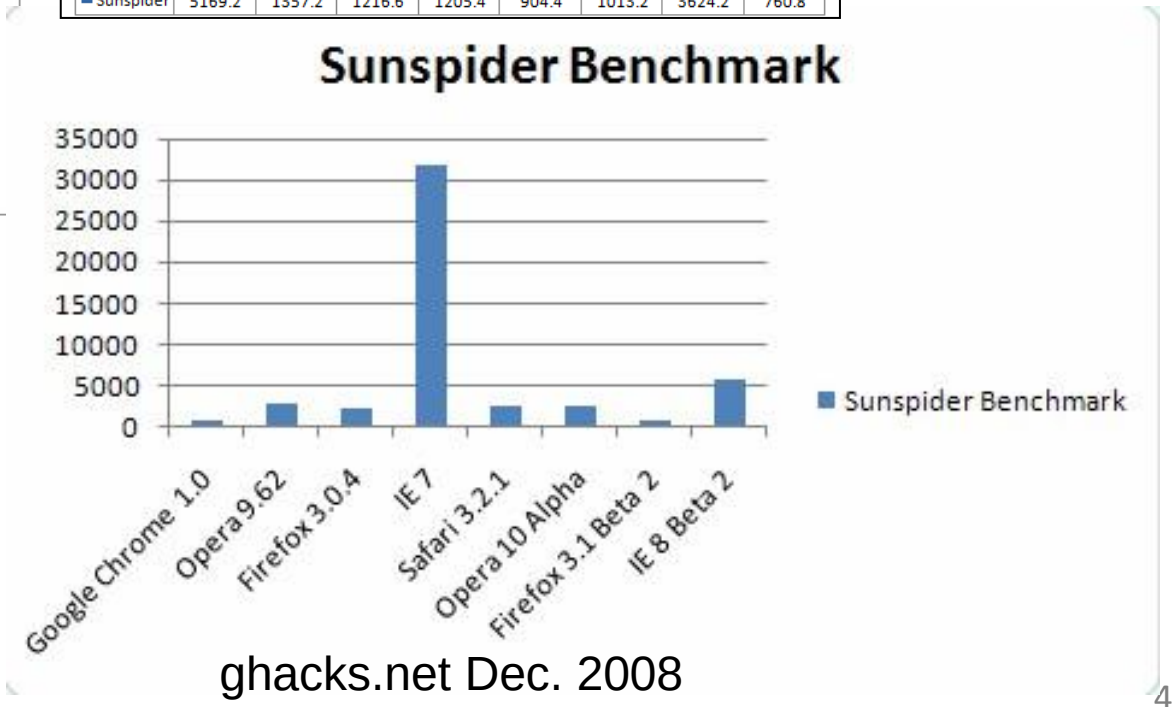
RiSE



ZDNet 29 May 2008



Browser Wars!



ghacks.net Dec. 2008

Microsoft Research

RiSE

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:

bing Google Gmail Windows Live Hotmail amazon.com ebay

Goals of JSMeter Project

- Instrument JavaScript execution and measure behavior
- Compare behavior of JavaScript benchmarks against real sites
- Consider how benchmarks can mislead design decisions

Microsoft Research

How We Measured JavaScript

\ie\jscript*.cpp

Source-level instrumentation

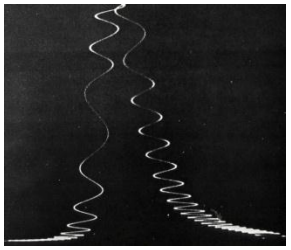
custom jscript.dll

---

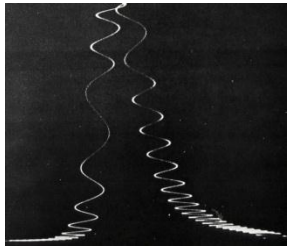
ebay

CNN

website visits

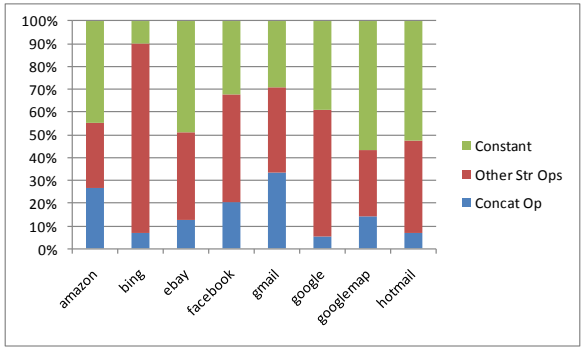


custom trace files



custom trace files

Offline analyzers



Microsoft Research

RiSE

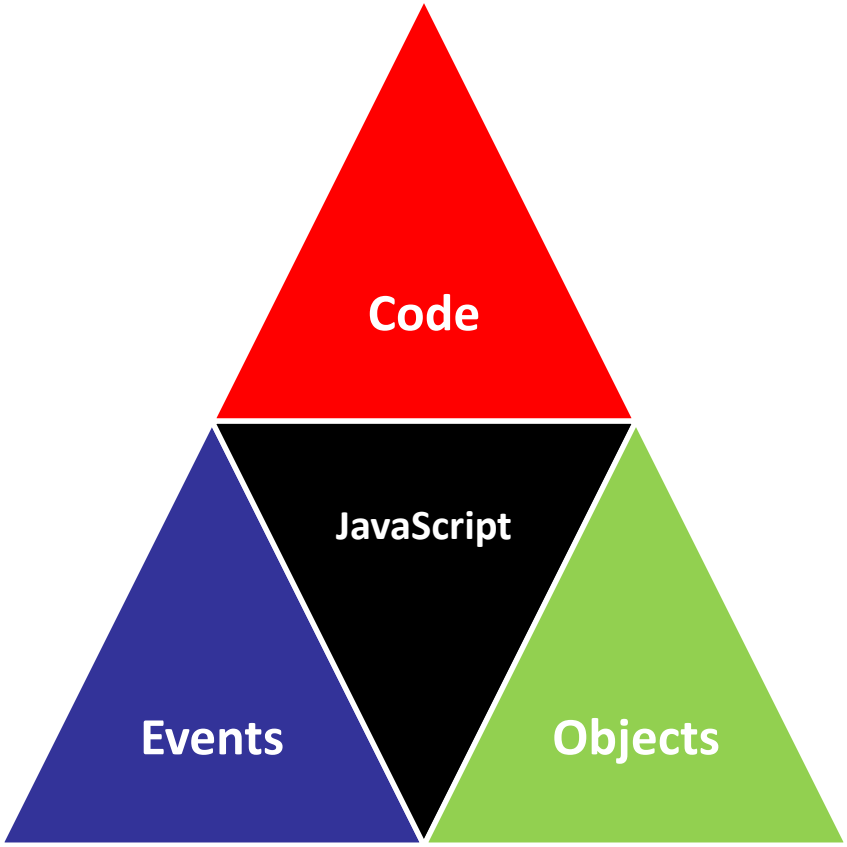
Visiting the Real Sites

- 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, sing 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

RiSE

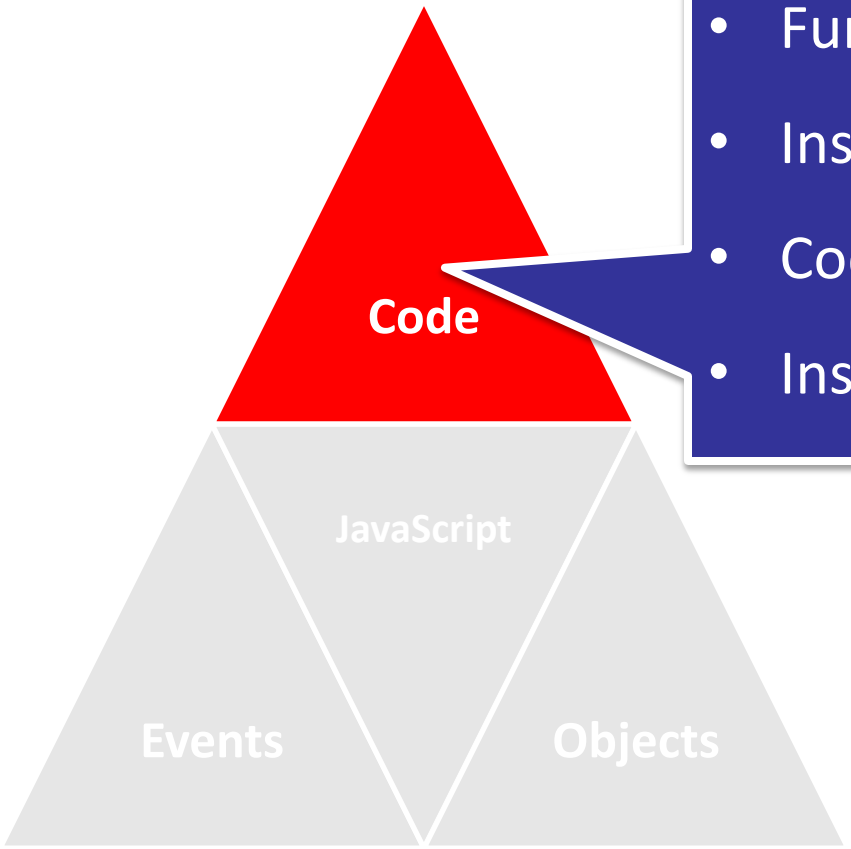
Understanding JavaScript Behavior



Microsoft Research

RiSE

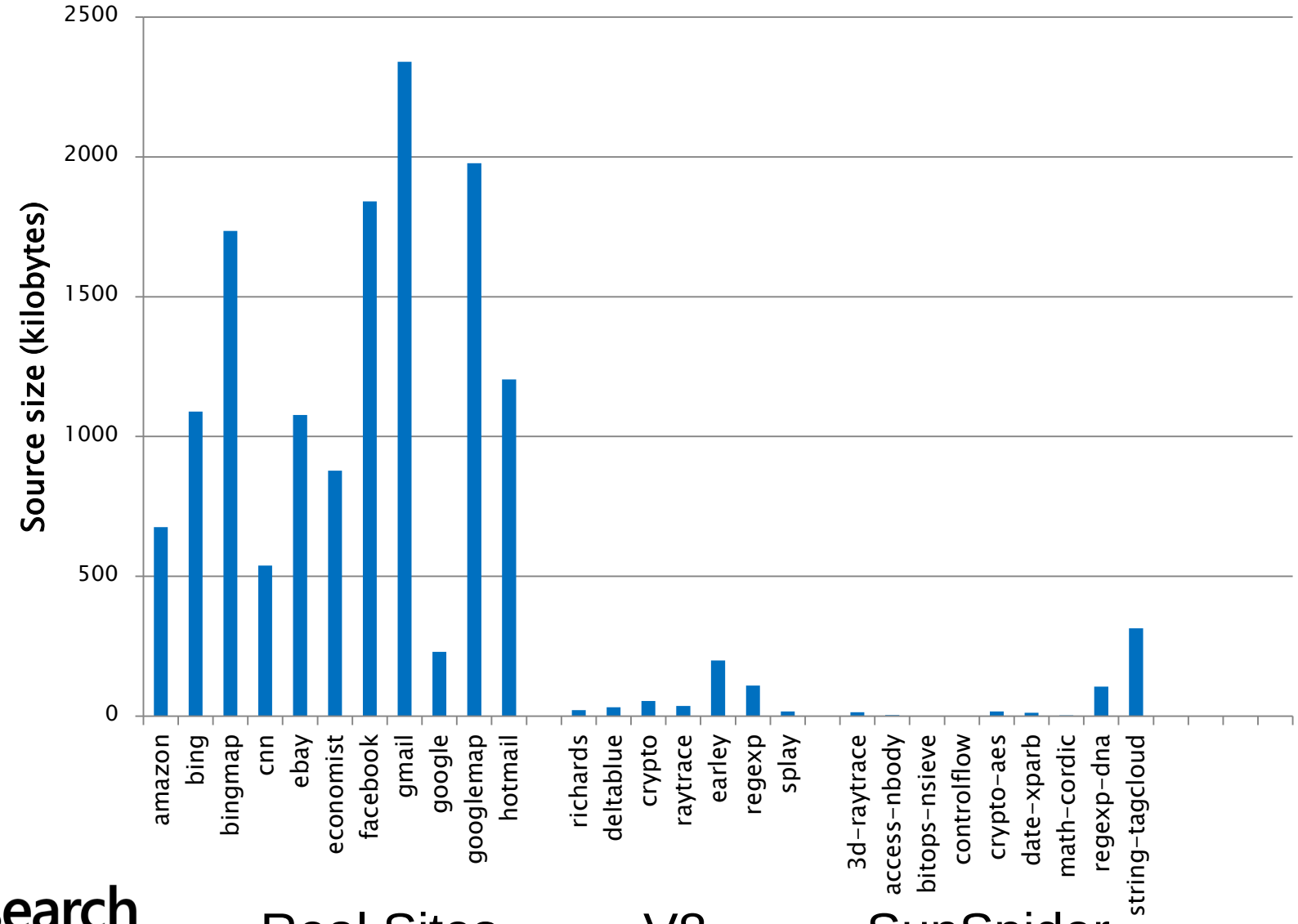
Code Behavior



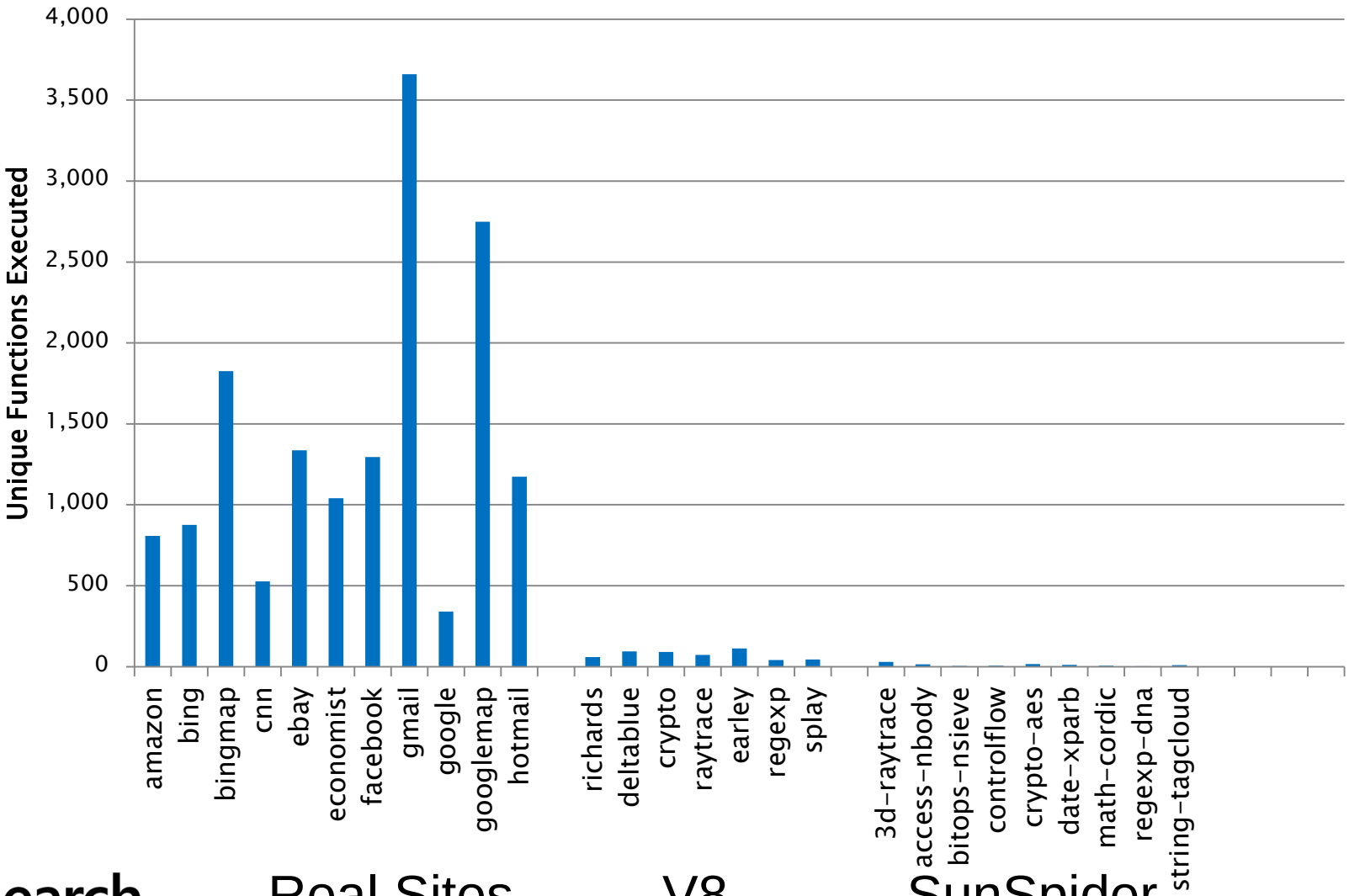
- Function size
- Instructions/call
- Code locality
- Instruction mix

Microsoft Research

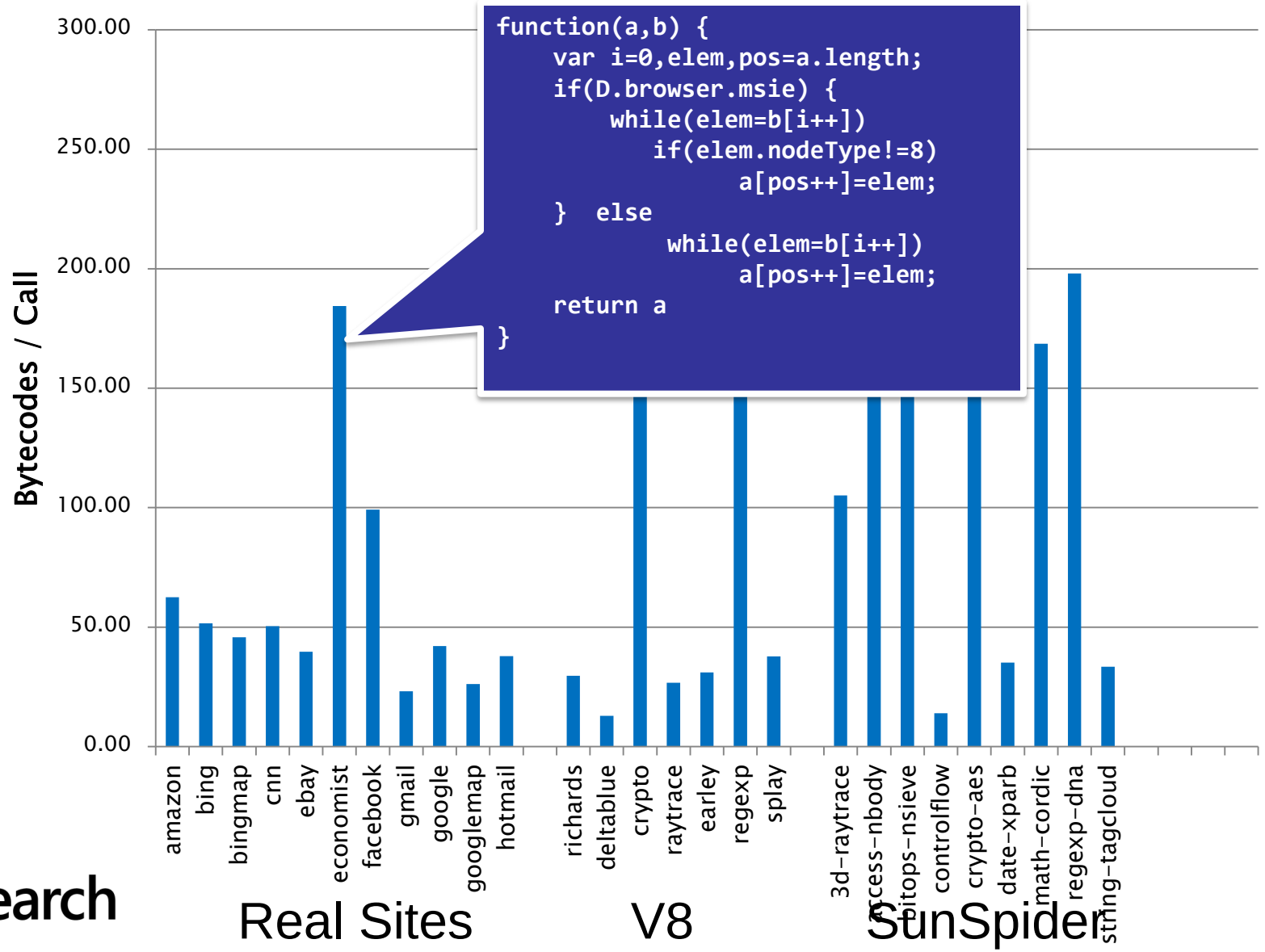
Total Bytes of JavaScript Source



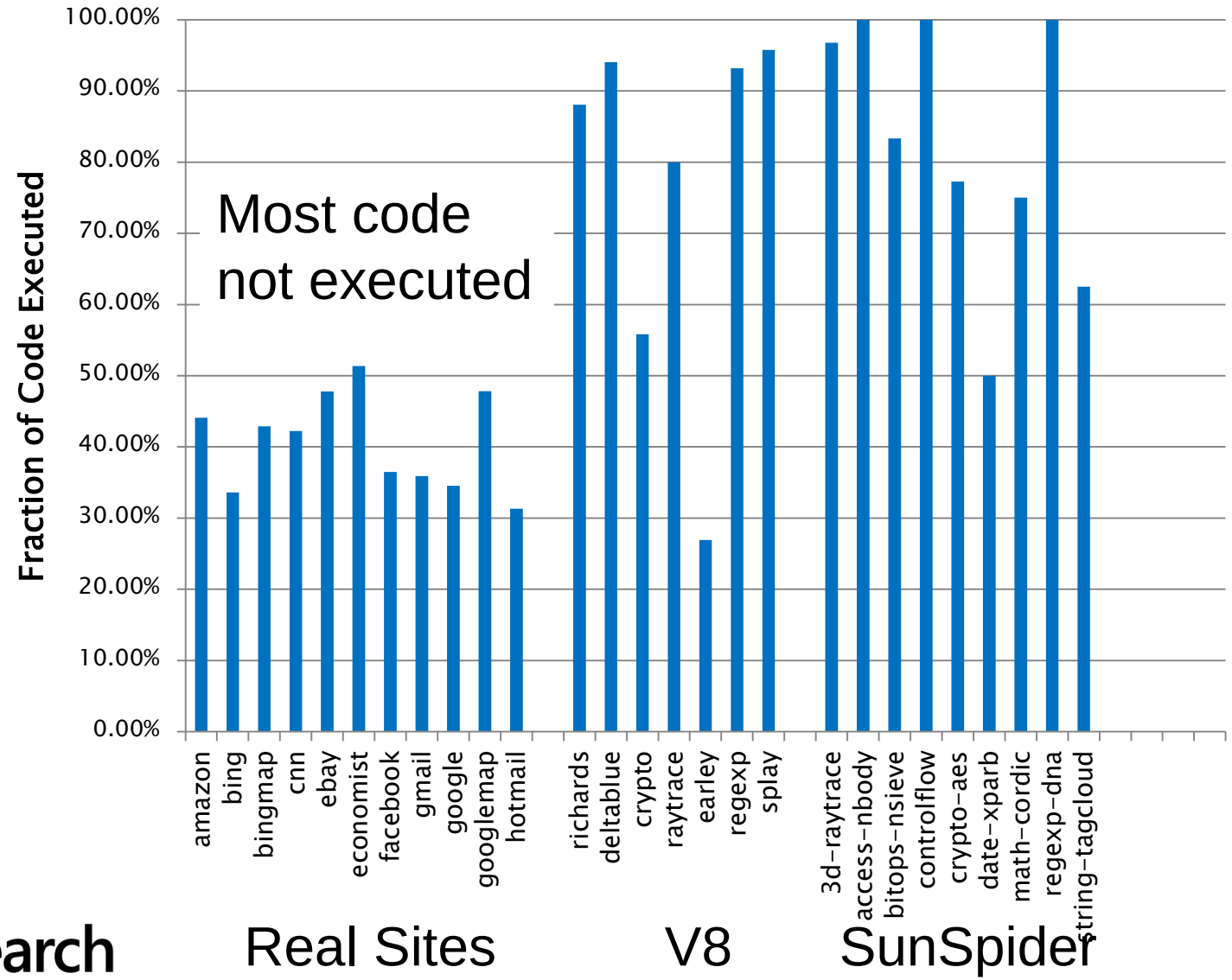
Static Unique Functions Executed



Bytecodes / Call

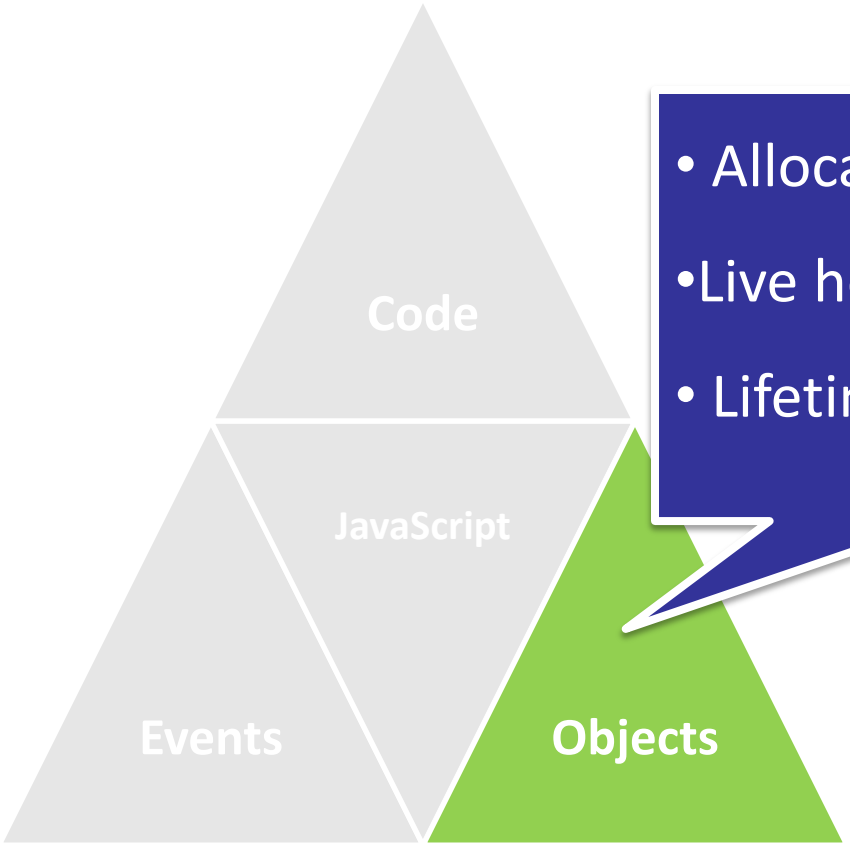


Fraction of Code Executed



RiSE

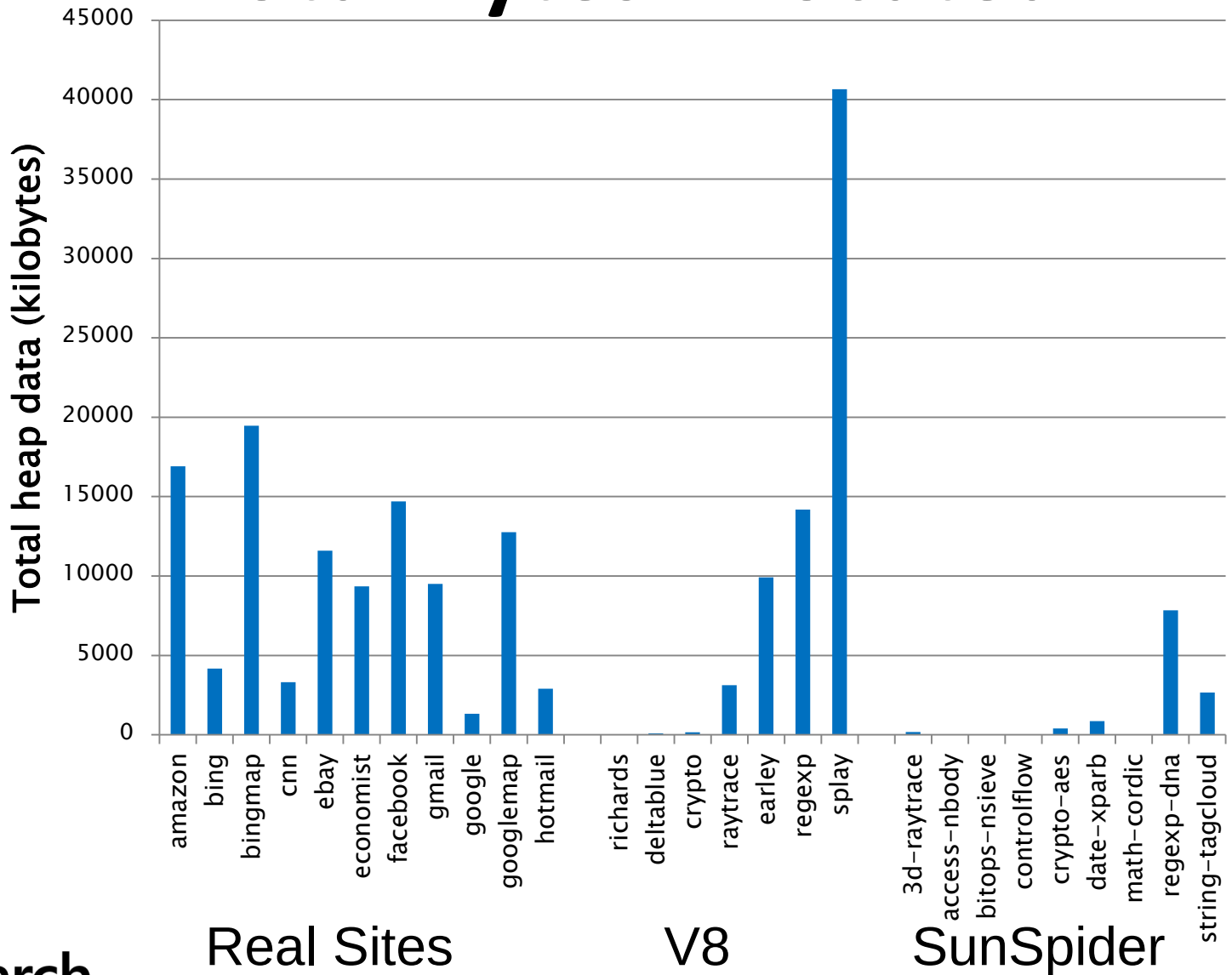
Object Allocation Behavior



- Allocation by types
- Live heap composition
- Lifetime distribution

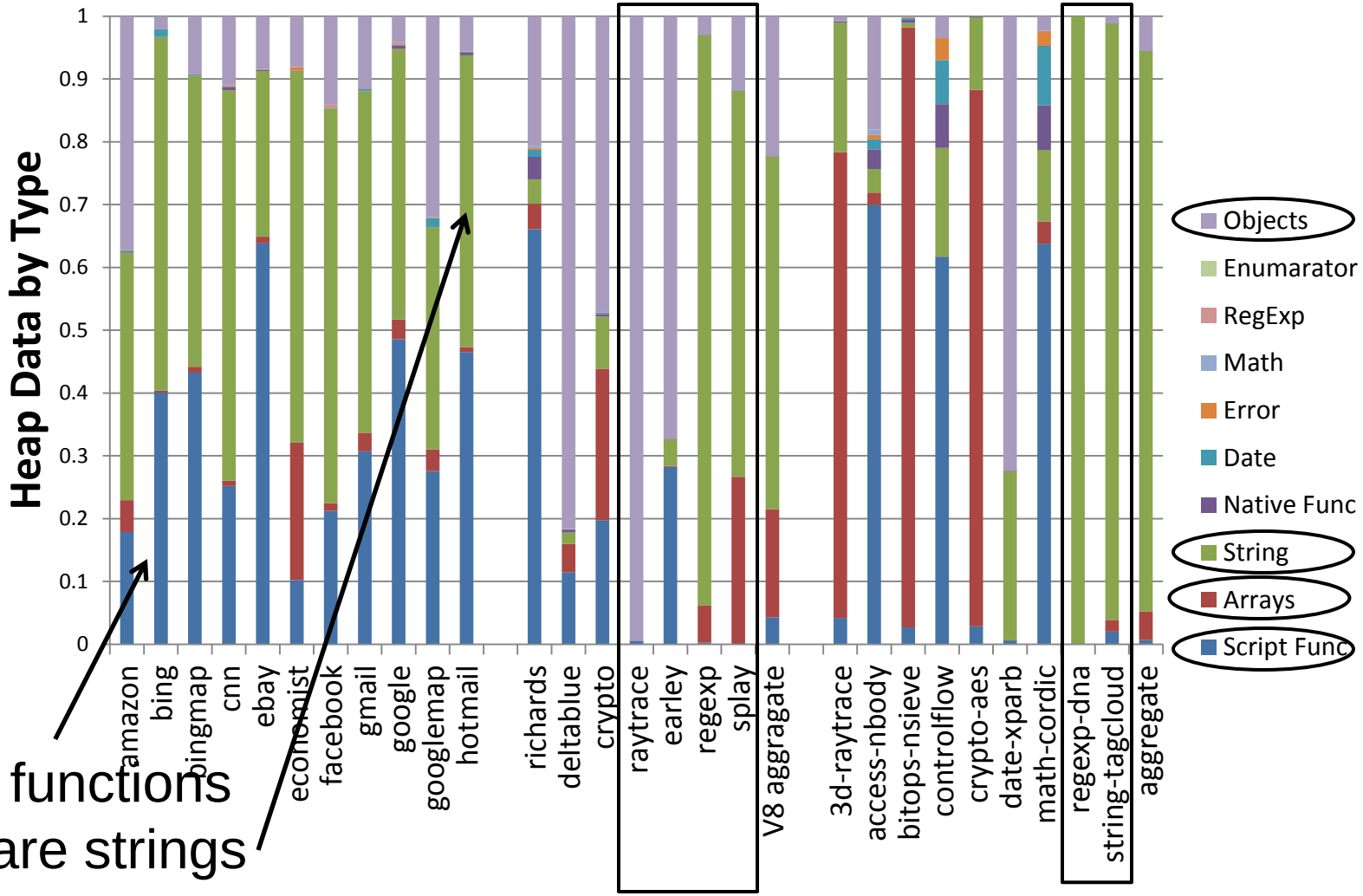
Microsoft Research

Total Bytes Allocated



Heap Data by Type

Few benchmarks allocate much data



Many functions

Rest are strings

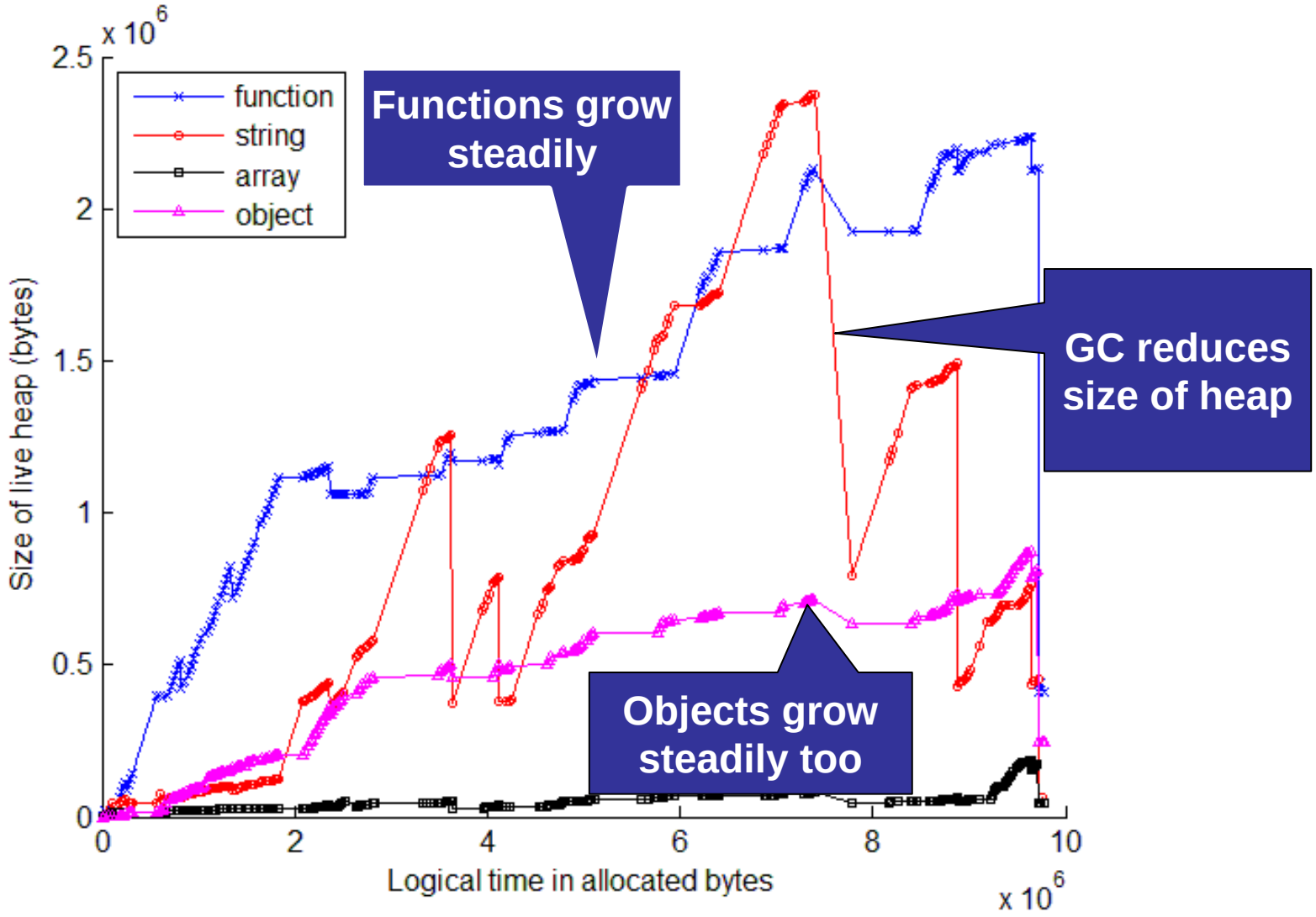
Real Sites

V8

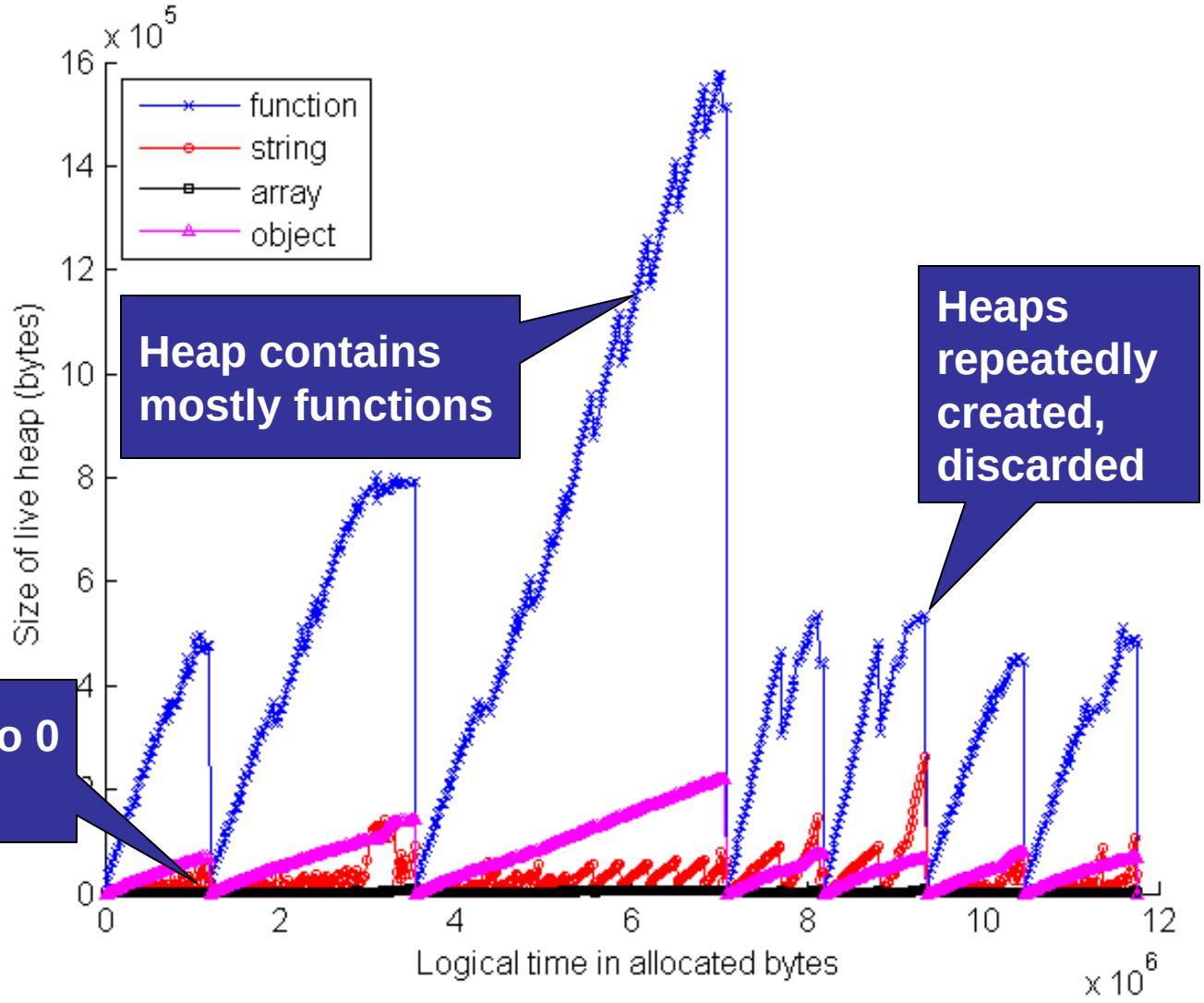
SunSpider

Microsoft Research

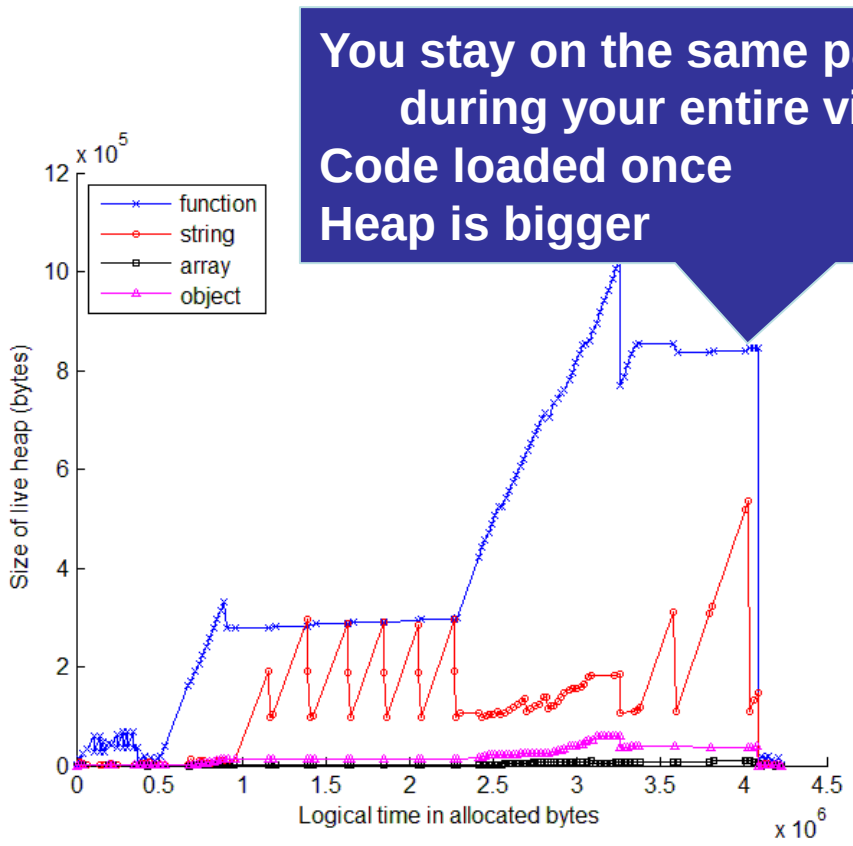
Live Heap Over Time (gmail)



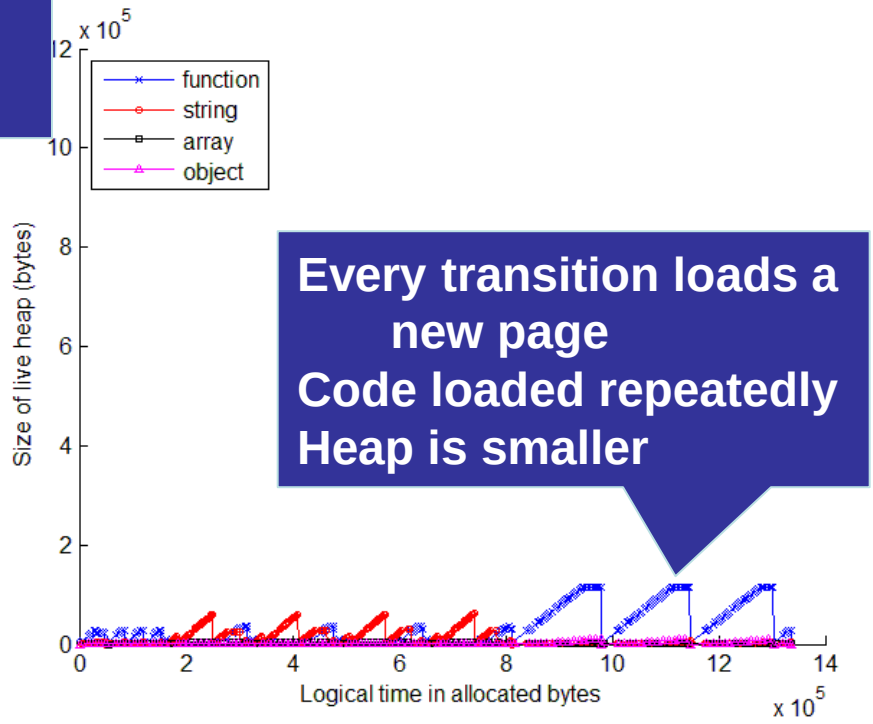
Live Heap over Time (ebay)



2 Search Websites, 2 Architectures



Bing

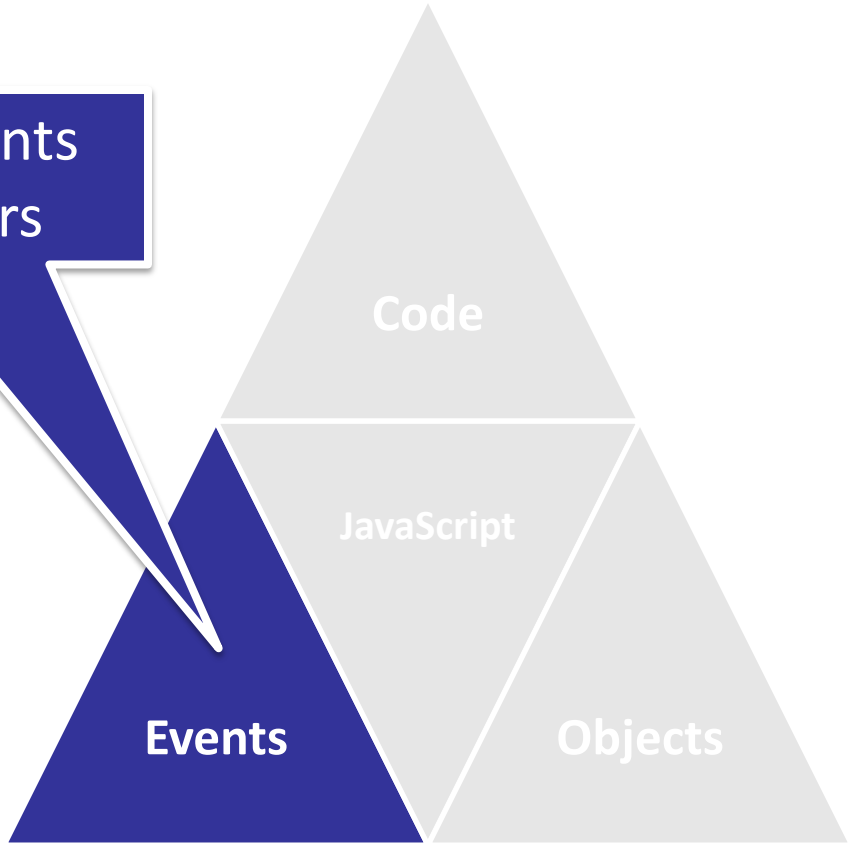


Google

RiSE

Event Handlers in JavaScript

- Number of events
- Sizes of handlers

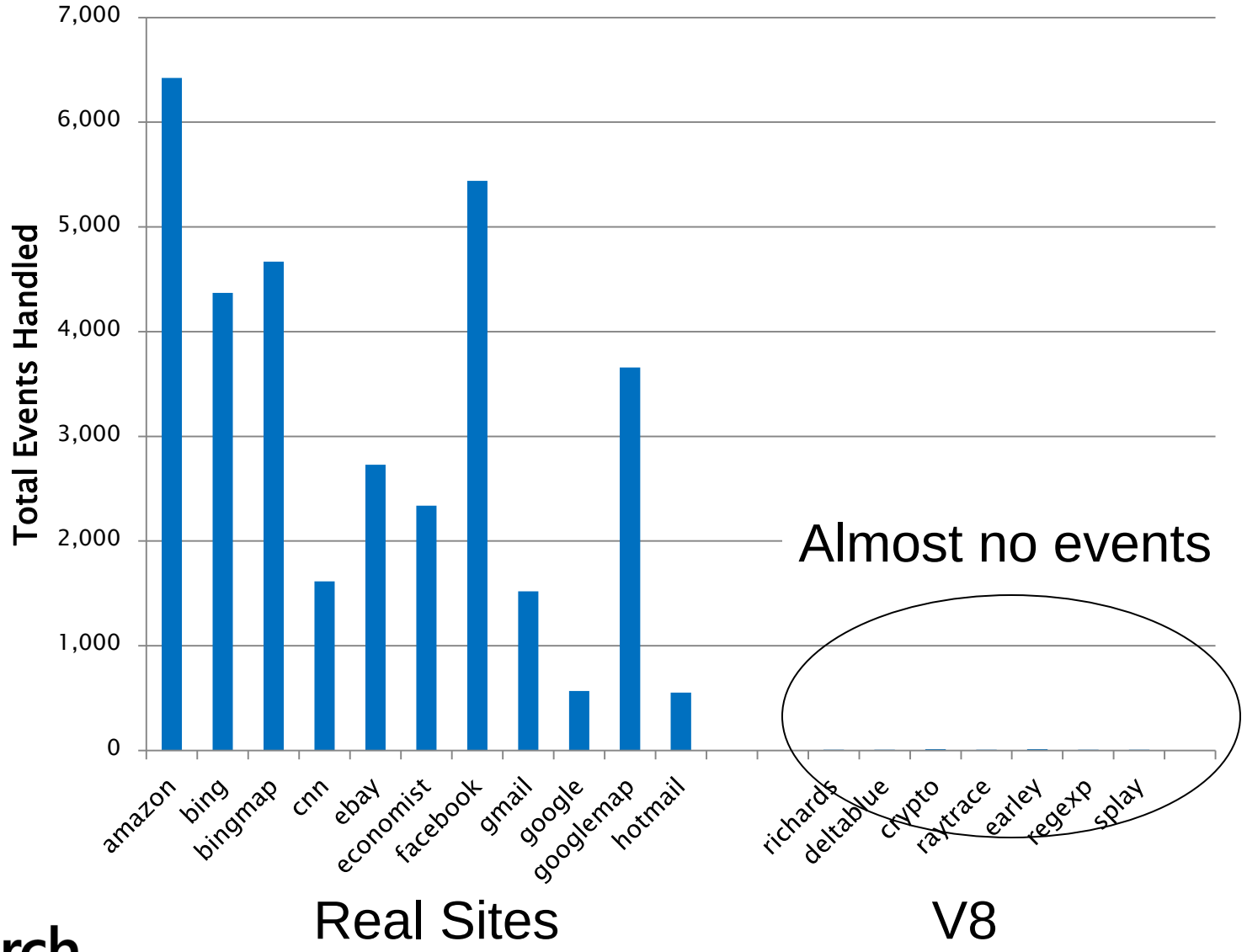


Microsoft Research

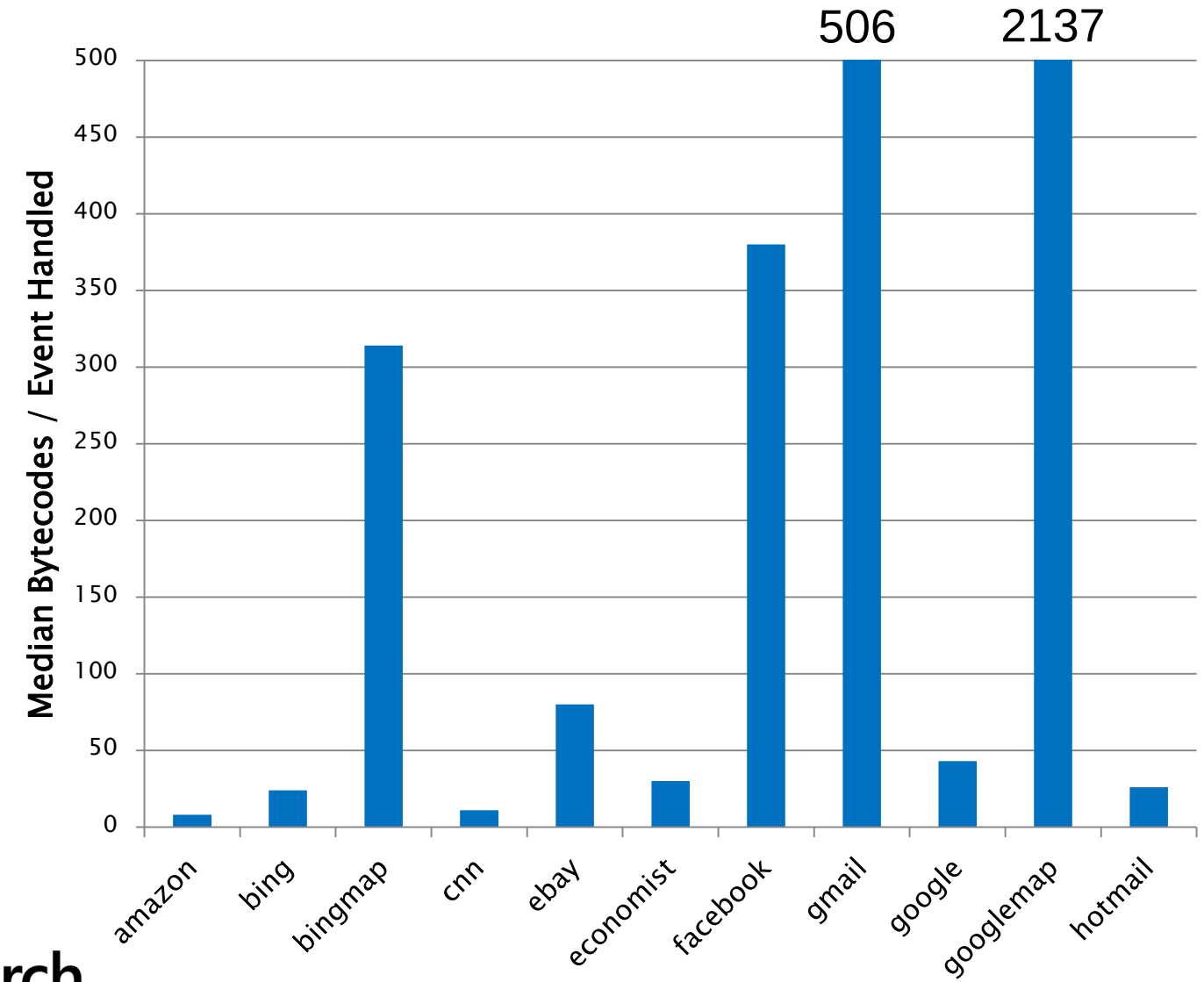
Event-driven Programming Model

- 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



RISE

Sure, this is all good, but…

- Everyone knows benchmarks are unrepresentative
- How much difference does it make, anyway?
- Wouldn't any benchmarks have similar issues?

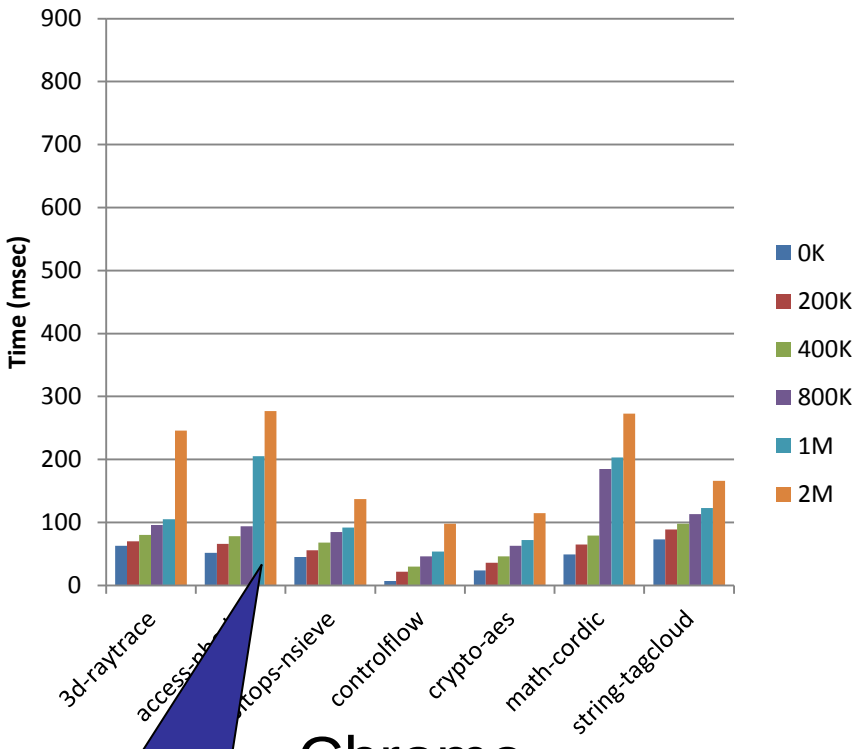
RiSE

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 7 SunSpider benchmarks
  - We measured the impact on the benchmark performance

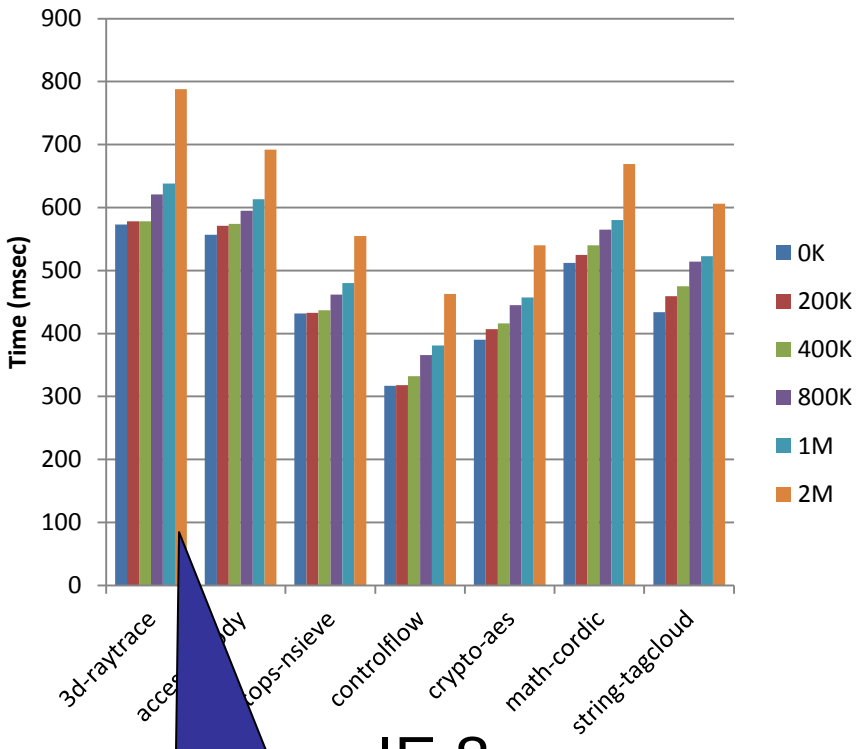
RiSE

Performance Impact of Cold Code



Chrome

Cold code makes SunSpider on Chrome up to 4.5x slower



IE 8

Cold code has non-uniform impact on execution time

Microsoft Research

RiSE

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

Microsoft Research

RiSE

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

Microsoft Research

RiSE

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
- **Paper:**
  - "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.
- **Additional measurements:**
  - "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.

Microsoft Research

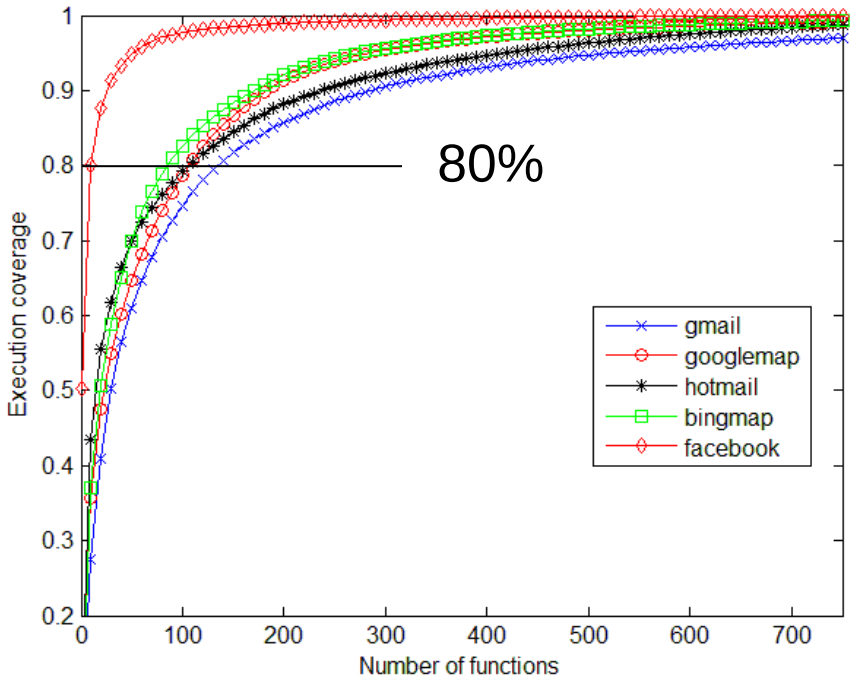
RiSE

Additional Slides

Microsoft Research

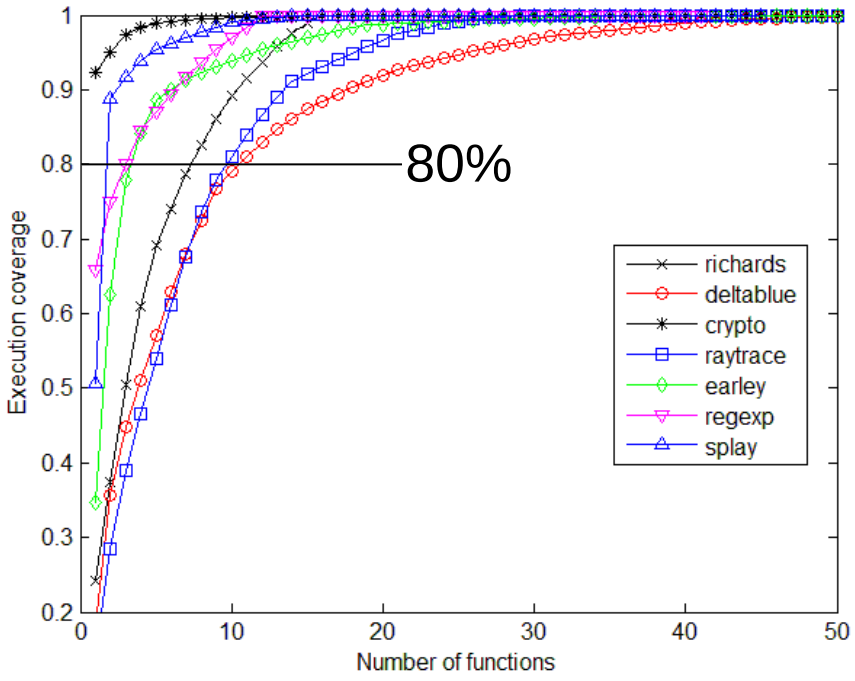
Hot Function Distribution

80% of time in 100+ functions



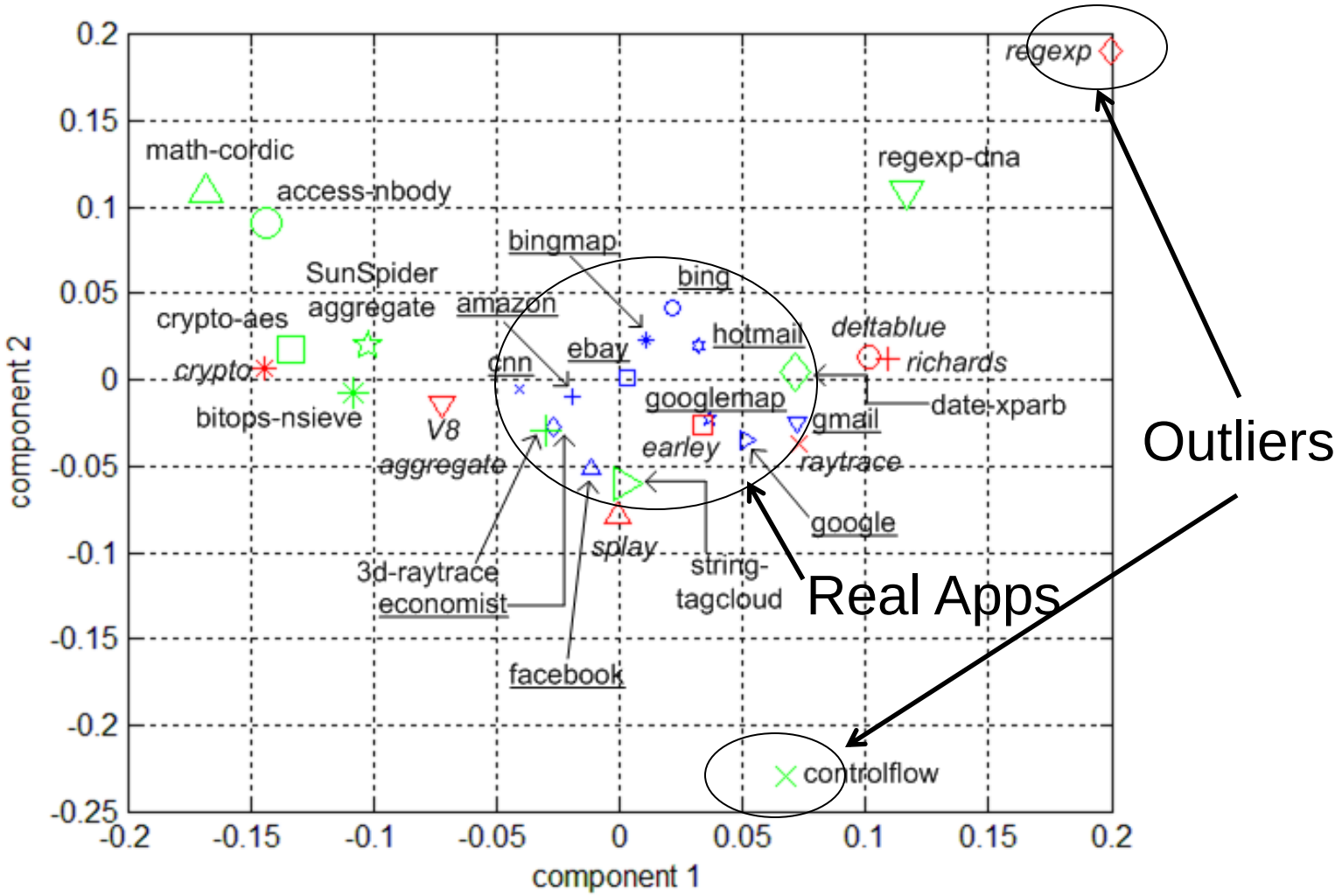
Real Sites

80% of time in < 10 functions



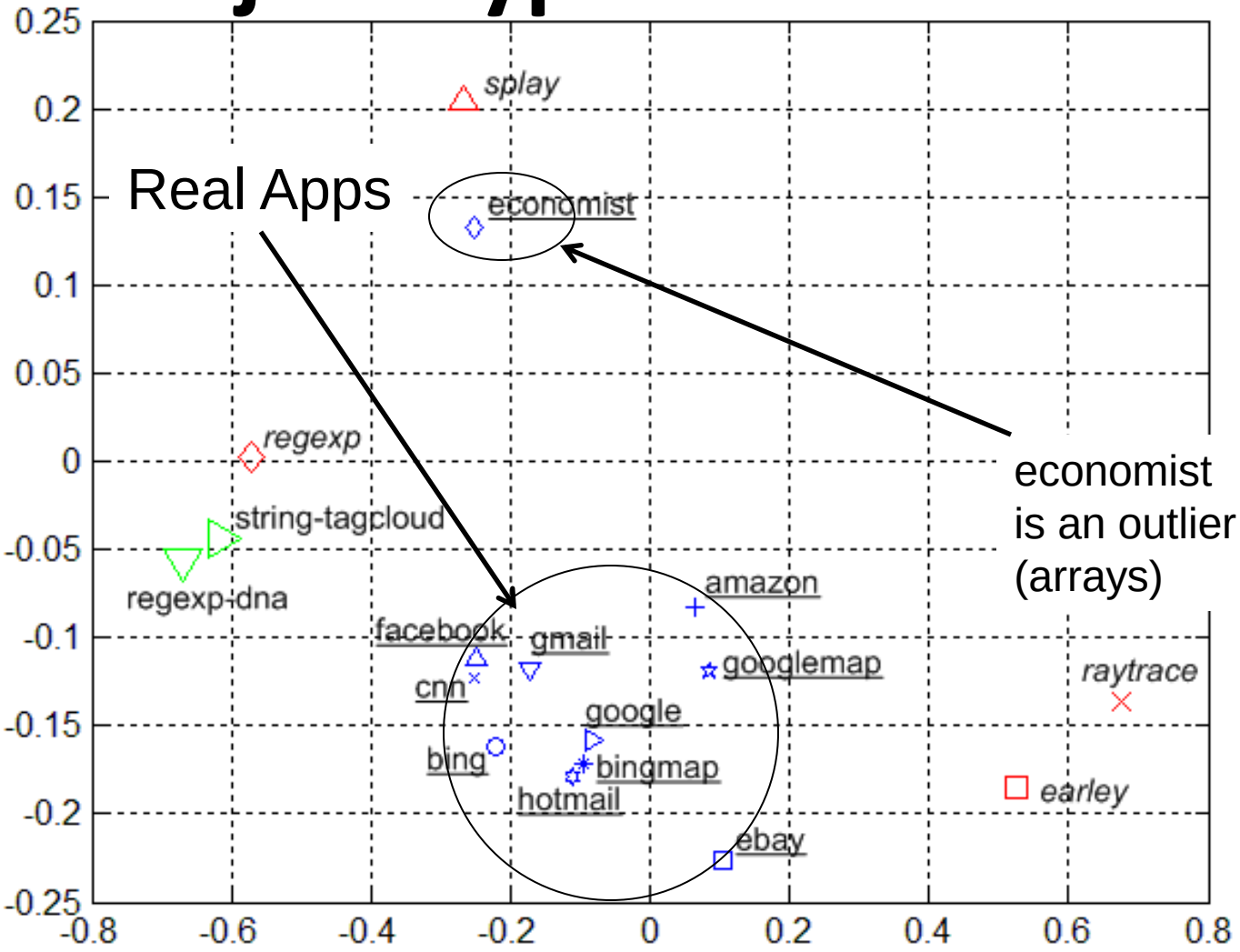
V8 Benchmarks

Opcode Distribution



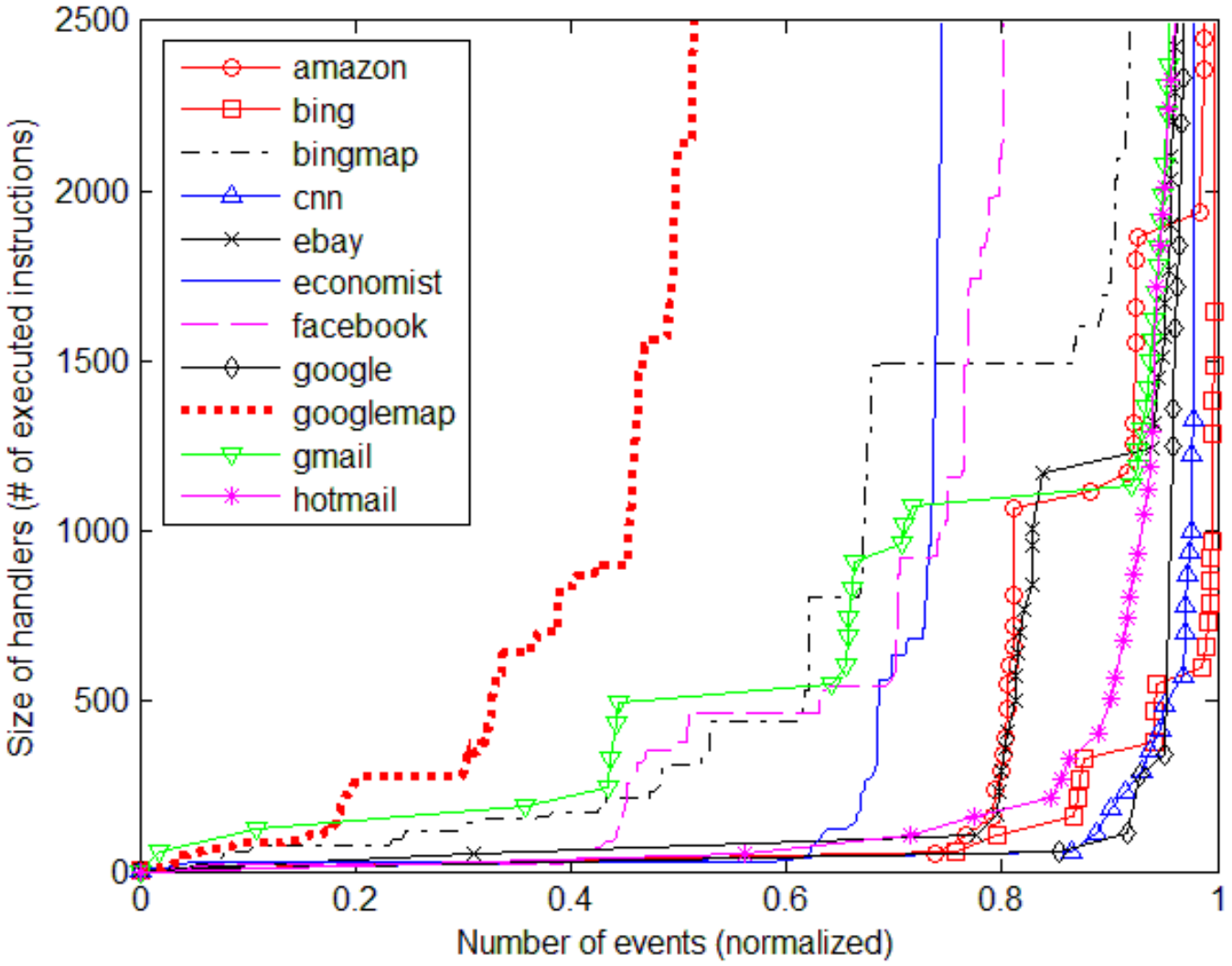
Green = SunSpider | Blue= Real Web Apps | Red = V8

Object Type Distribution



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

Distribution of Time in Handlers



Related Work

- JavaScript
  - Richards, Lebresne, Burg, and Vitek (PLDI'10)
  - Draw similar conclusions
- Java
  - Doufour et al. (OOPSLA'03), Dieckmann and U. Hölzle (ECOOP'99)
- Other languages
  - C++: Calder et al. (JPL'95)
  - Interpreted languages: Romer et al. (ASPLOS'96)