RiSE

RiSE Research in Software Engineering

Performance is Dead, Long Live Performance

Ben Zorn

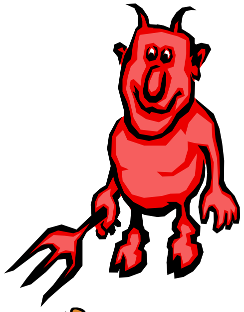
Microsoft Research

Microsoft Research

Outline

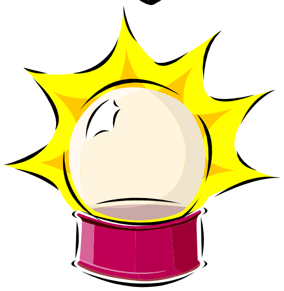
Good news

Bad news



Good news again!

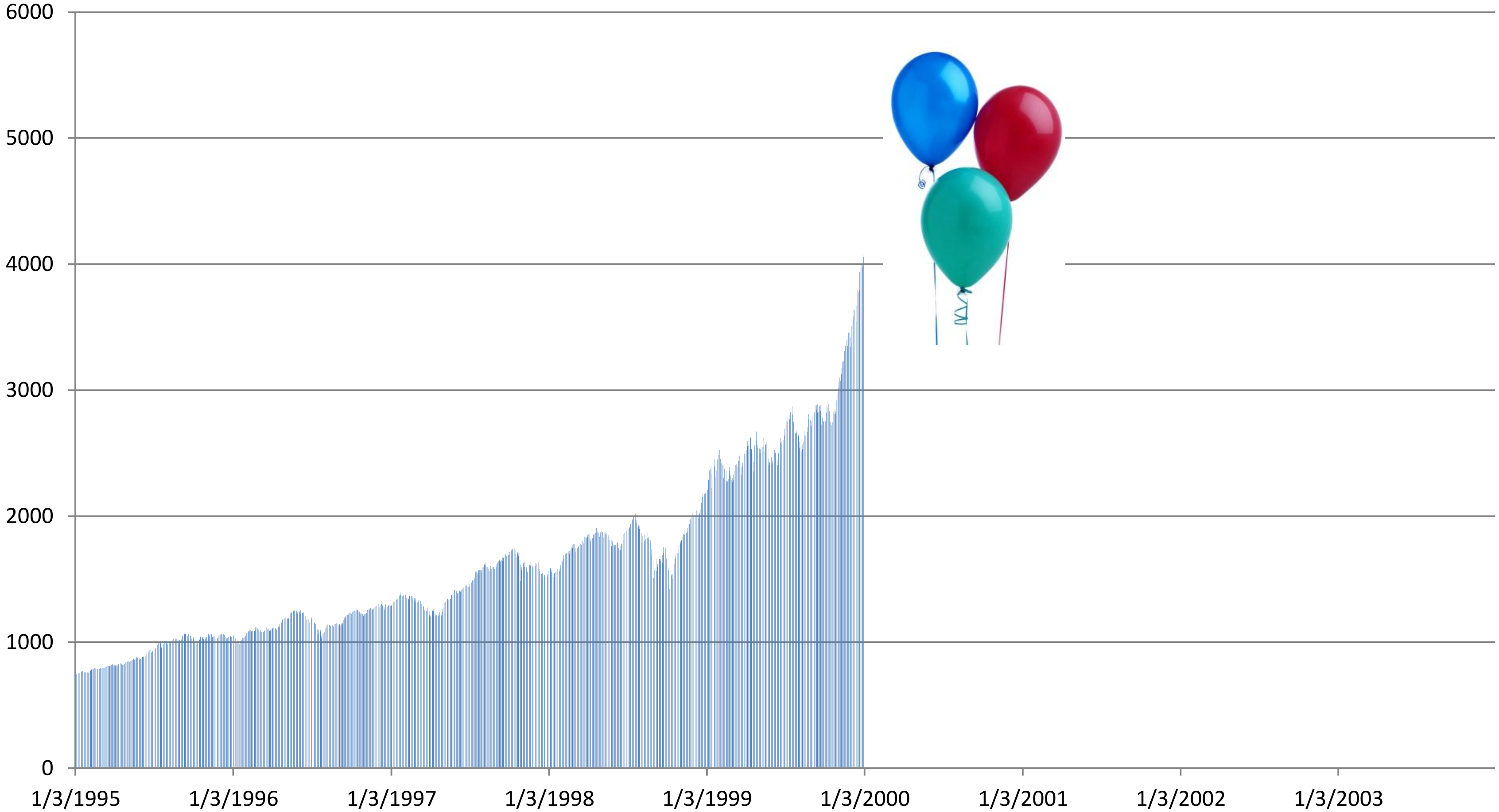
Mystery…



1990s
A Great Decade for Performance!

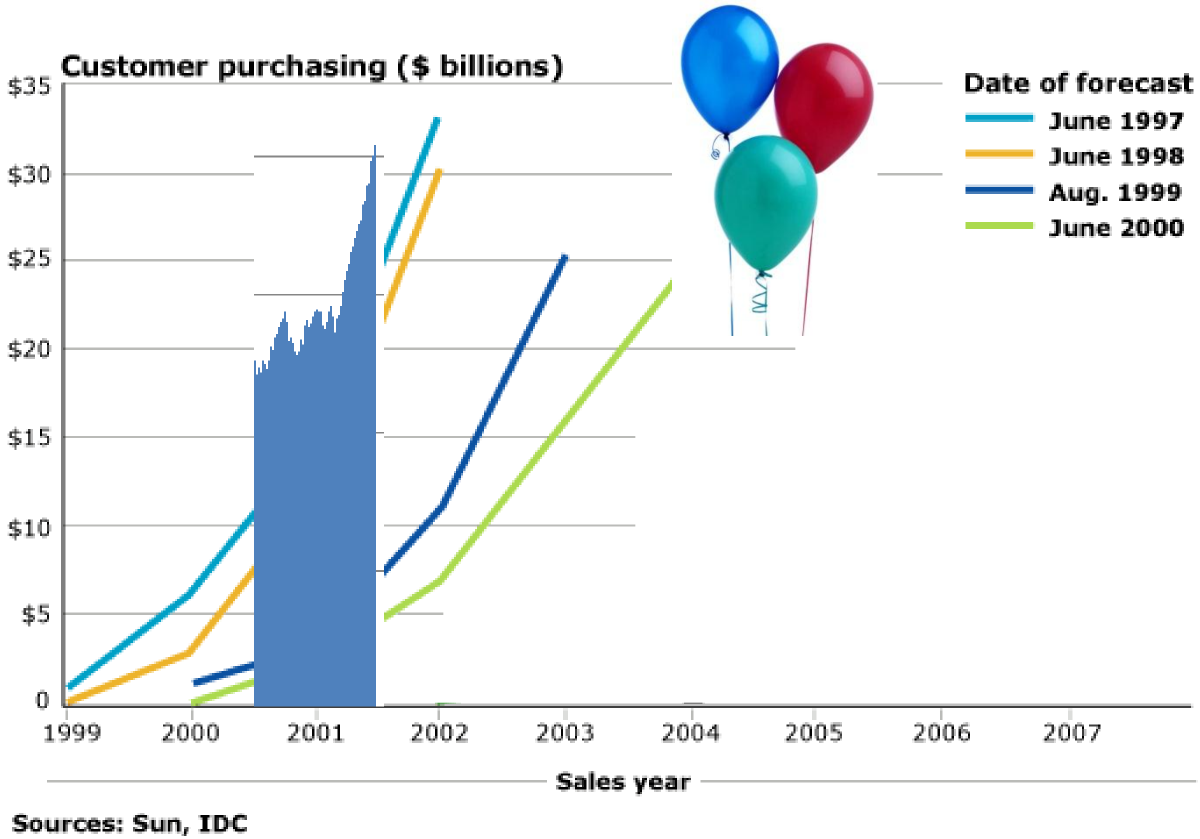
- Stock market booming
- Itanium processor shipping
- Processor performance growing exponentially (Moore's Law)
- Compiler research booming

NASDAQ Booming



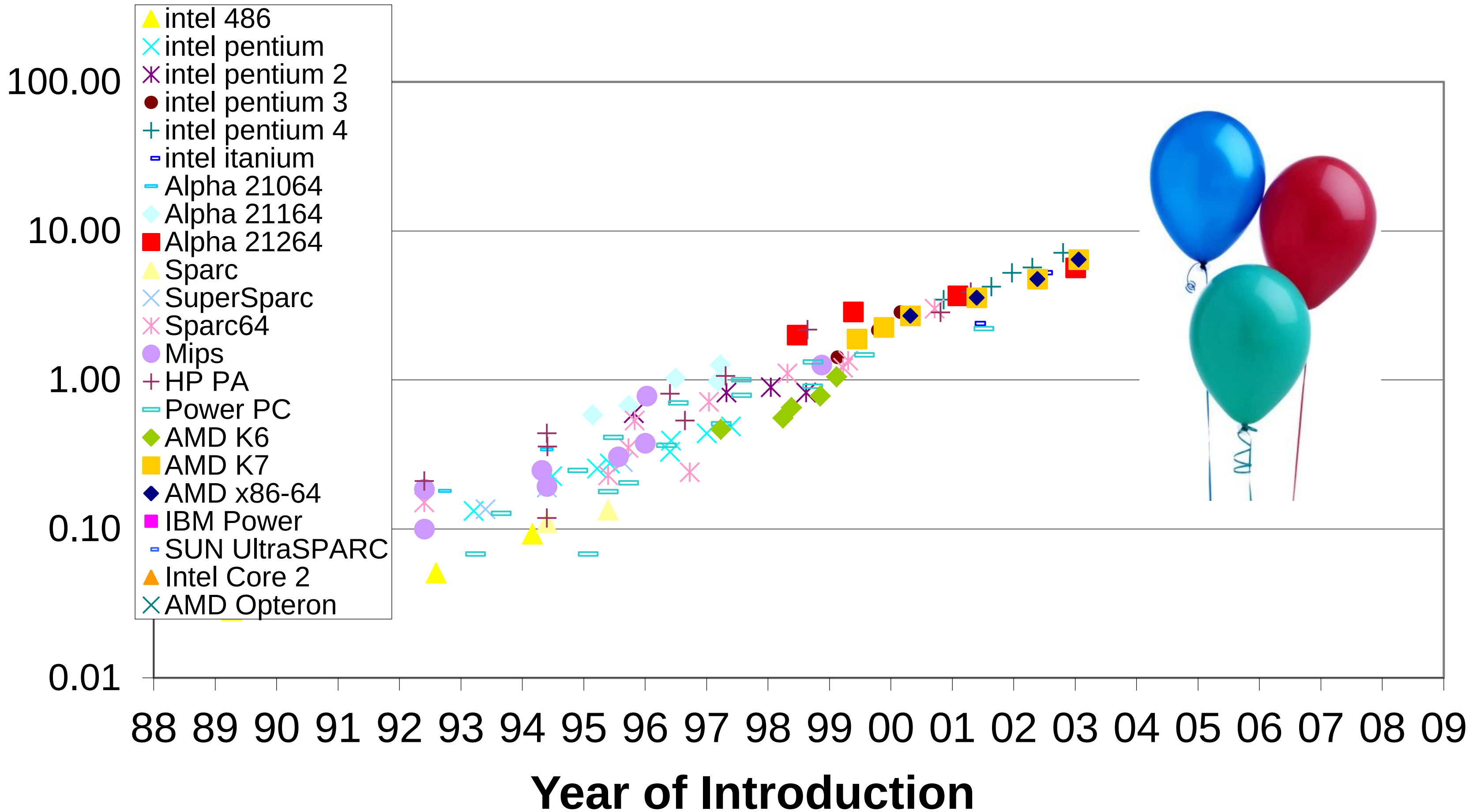
New Processors Had High Expectations

Itanium Sales Forecasts



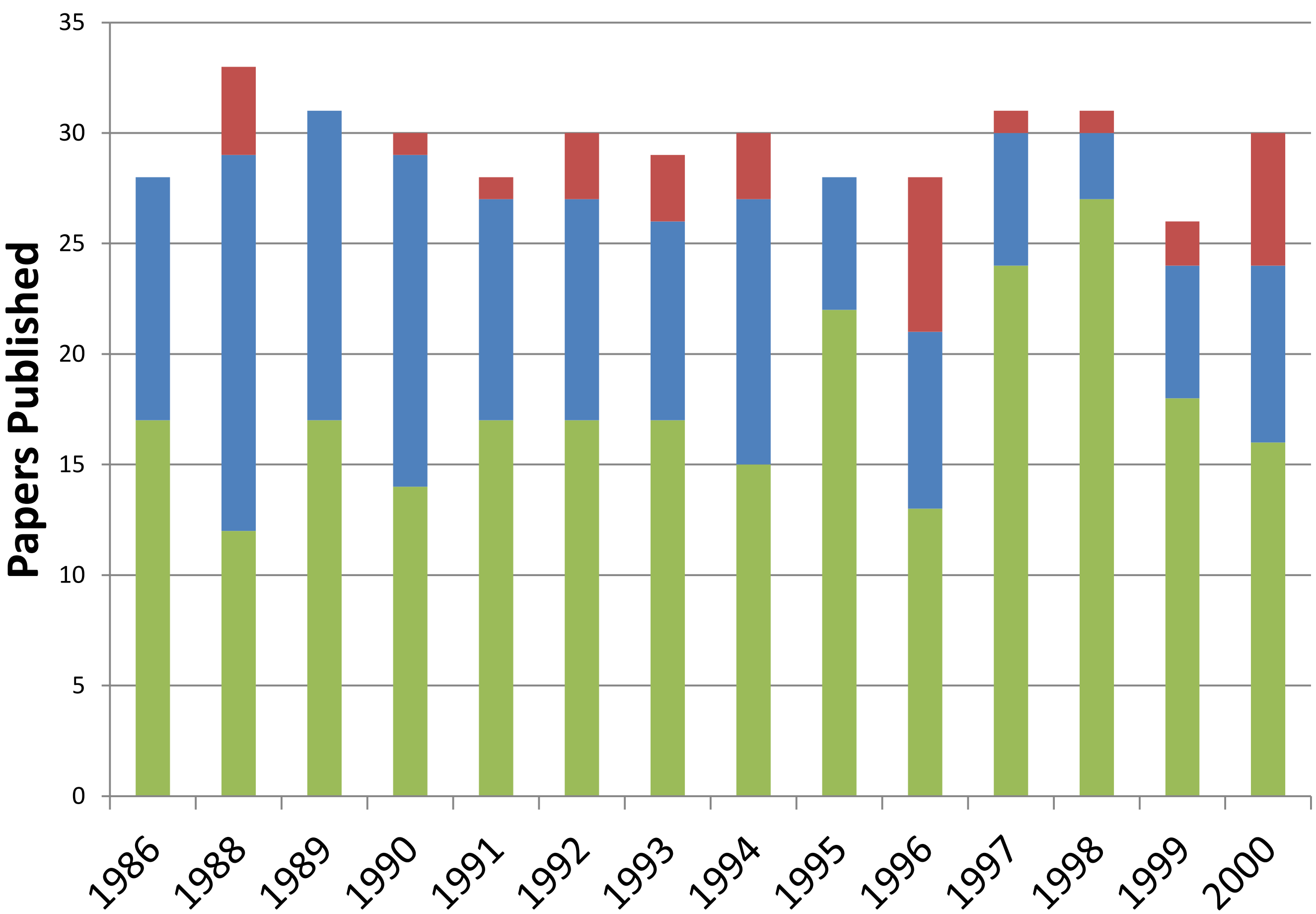
Source: CNET Networks from data provided by Sun and IDC (12/7/2005)

SPECint2006 CPU Performance



Numbers courtesy of Mark Horowitz, Ofer Shacham

Performance Papers Dominate PLDI



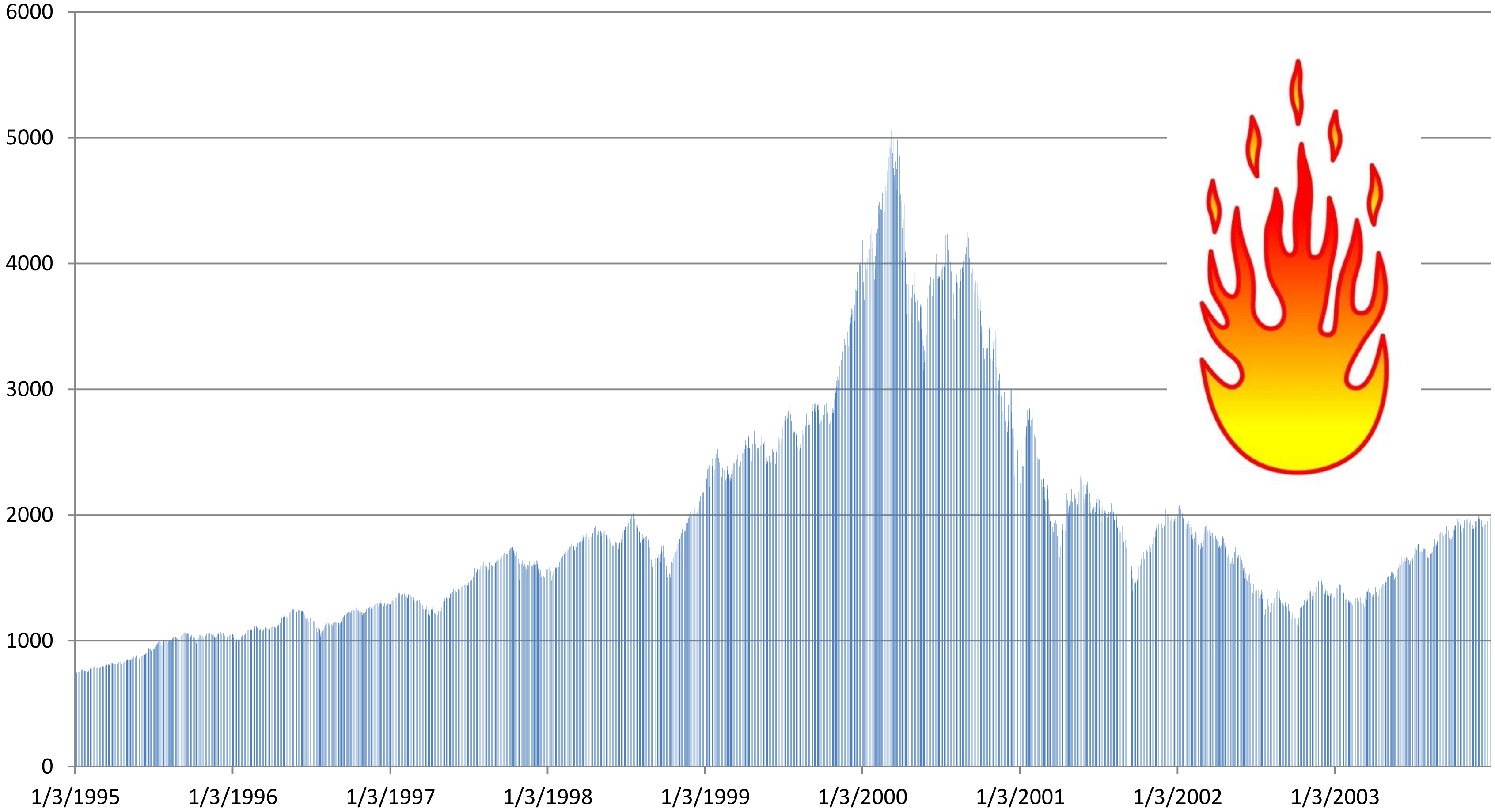
Some Cynics: Proebsting's Law

- **Proebsting's Law: Compiler Advances Double Computing Power Every 18 *Years***

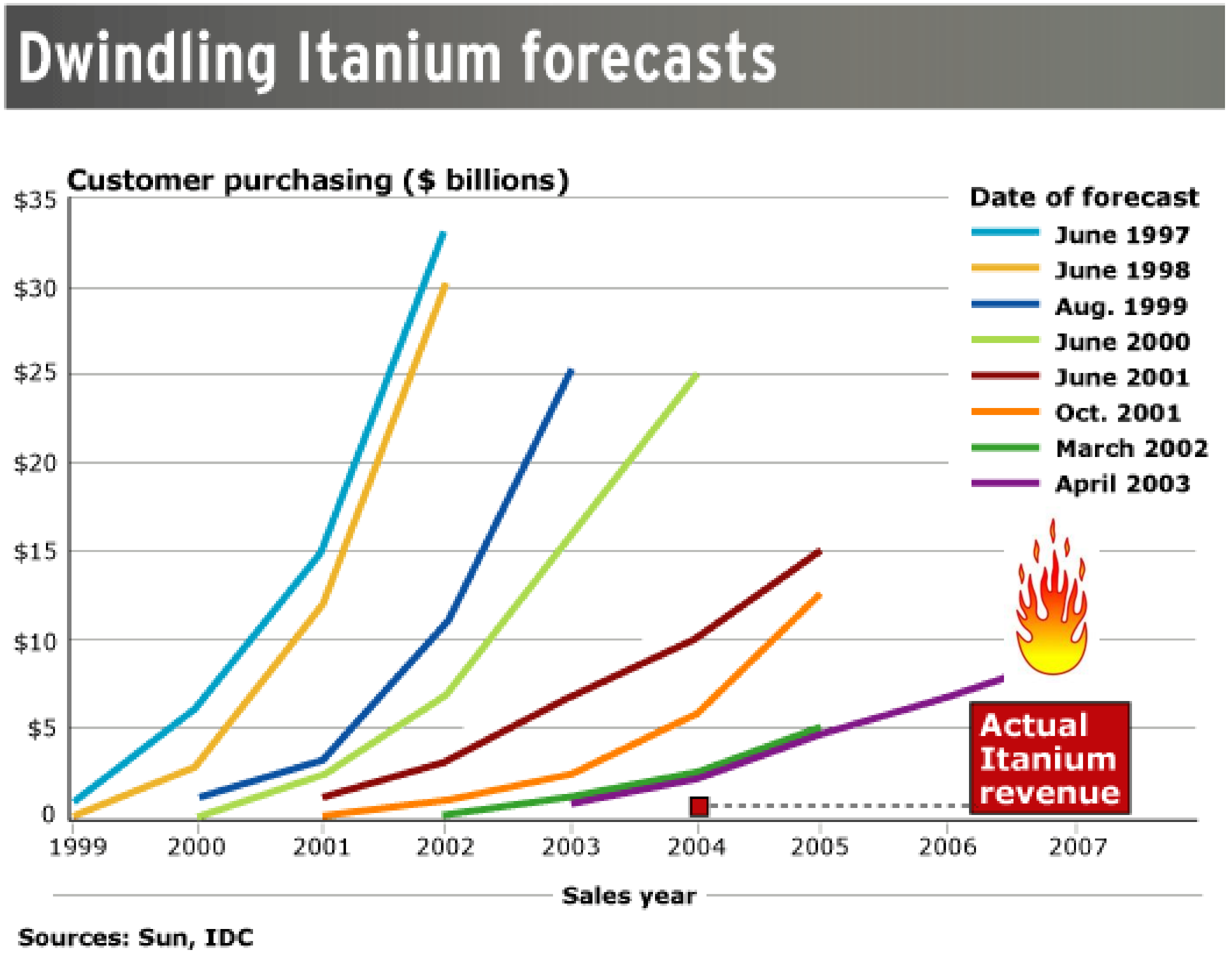
  *"…*This means that while hardware computing horsepower increases at roughly 60%/year, compiler optimizations contribute only 4%. Basically, compiler optimization work makes only marginal contributions."

http://research.microsoft.com/en-us/um/people/toddpro/papers/law.htm

The Bubble Bursts



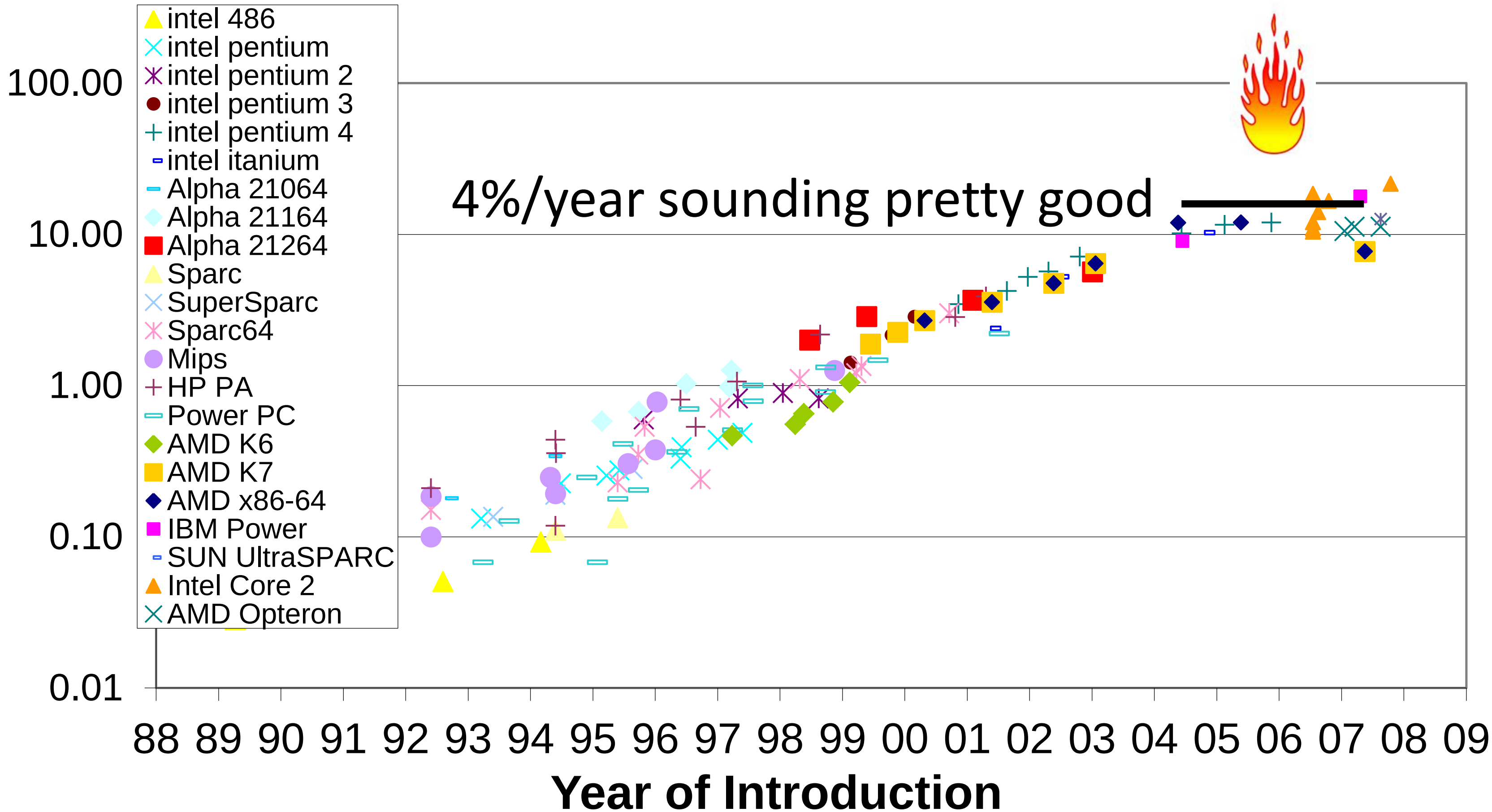
Itanium Sales Lag



Source: CNET Networks from data provided by Sun and IDC (12/7/2005)

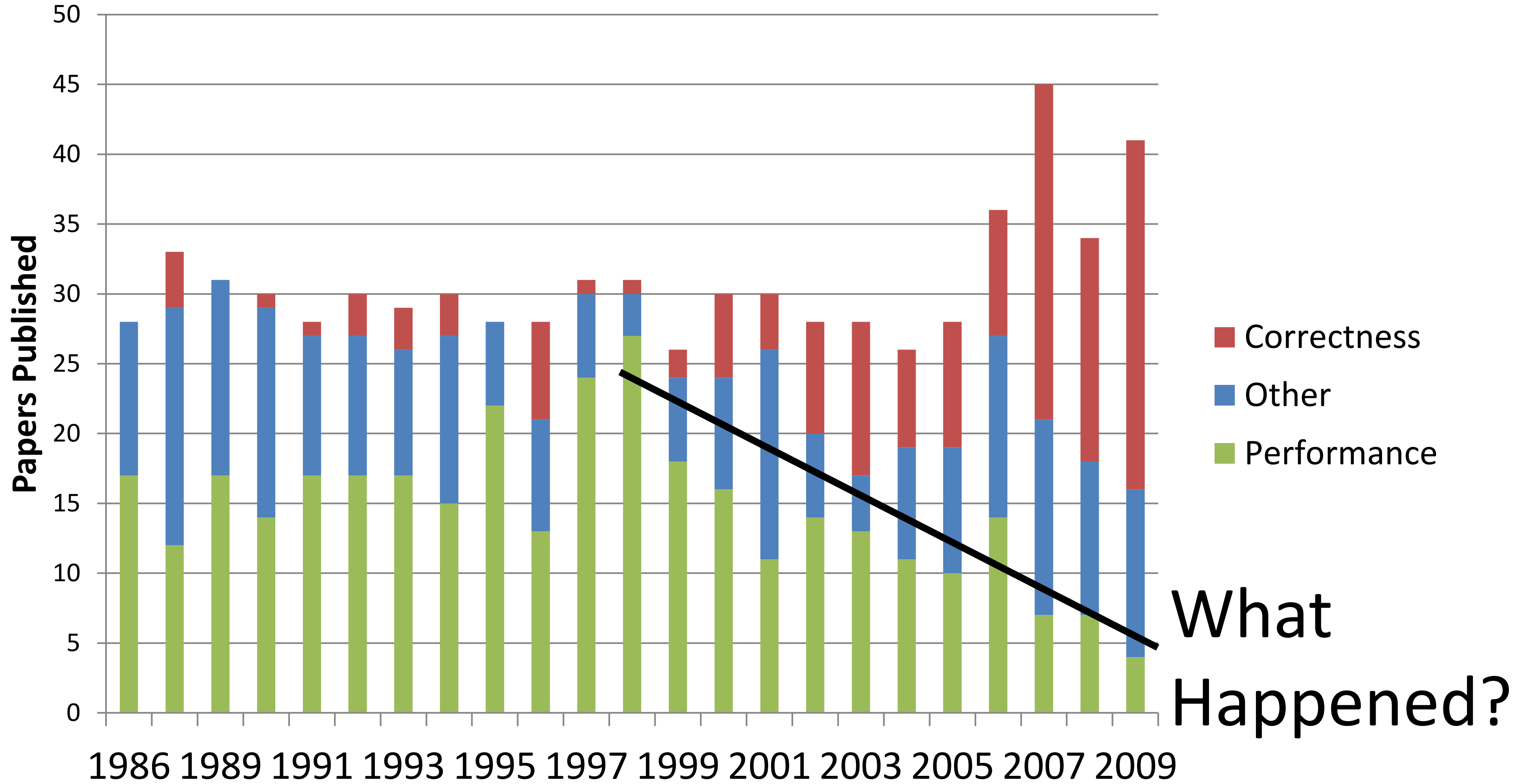
http://news.cnet.com/2300-1006_3-5873647.html

Uniprocessor Performance Flattens



Numbers courtesy of Mark Horowitz, Ofer Shacham

PLDI Performance Paper Decline



Performance is Dead

What Killed Performance?

Advisories

ANALYSIS: .ida "Code Red" Worm

Release Date:
July 17, 2001

Severity:
HIGH

Vendor:
Microsoft has previously released a patch for this .ida vulner

Code Red
July 2001
359k hosts, 1 day

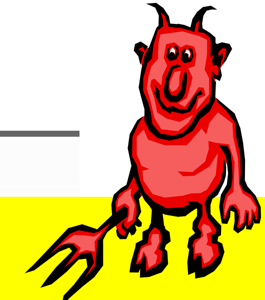
CERT® Advisory CA-2001-26 Nimda Worm

Original release date: September 18, 20
Revised: September 25, 2001
Source: CERT/CC

A complete revision history is at the en

Systems Affected

- Systems running Microsoft Windows 95, 98, ME, NT, and 2000



Nimda
September 2001
Became largest worm in 22 minutes

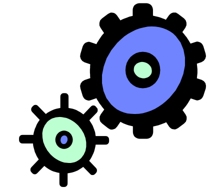
MS SQL WORM IS DESTROYING INTERNET BLOCK PORT 1434!

From: Michael Bacarella <mbac () netgraft com>
Date: Sat, 25 Jan 2003 02:11:41 -0500

```
I'm getting massive packet loss to various
I am seeing a lot of these in my tcpdump o
host.

02:06:31.017088 150.140.142.17.3047 > 24.1
02:06:31.017244 24.193.37.212 > 150.140.14                      reachable [tos 0x

It looks like there's a worm affecting MS
pingflooding addresses at some random sequence.

All admins with access to routers should block port 1434 (ms-sql-m)!

Everyone running MS SQL Server shut it the hell down or make
```

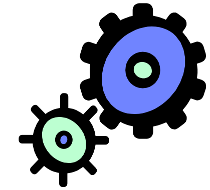
Slammer
January 2003
Infected 90% of vulnerable hosts in < 10 minutes

Companies Shift Gears



- Correctness and **security** a major new focus
- Microsoft investments:
  - PREfix, PREfast, SDV (Slam), ESP
  - Large code bases automatically checked for correctness errors (10+ million LOC)
- “Combined, the tools [PREfix and PREfast] found 12.5% of the bugs fixed in Windows Server 2003”
  - “Righting Software”, Larus et al., *IEEE Software,* 2004

Researchers Shift Gears

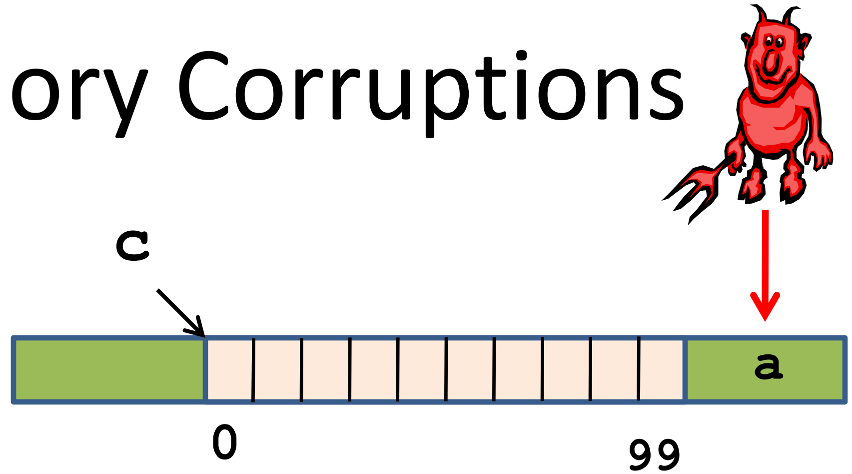


- Ben's research agenda changes
- 1990s
  - Predicting object lifetime and locality (with David Barrett and Matt Seidl)
  - Branch Prediction (with Brad Calder et al.)
  - Value Prediction (with Martin Burtscher)
- 2000s –tough sounding project names
  - DieHard – with Emery Berger, Gene Novark
  - Samurai – with Karthik Pattabiraman
  - Nozzle – with Ben Livshits

The New Threat: Exploitable Memory Corruptions

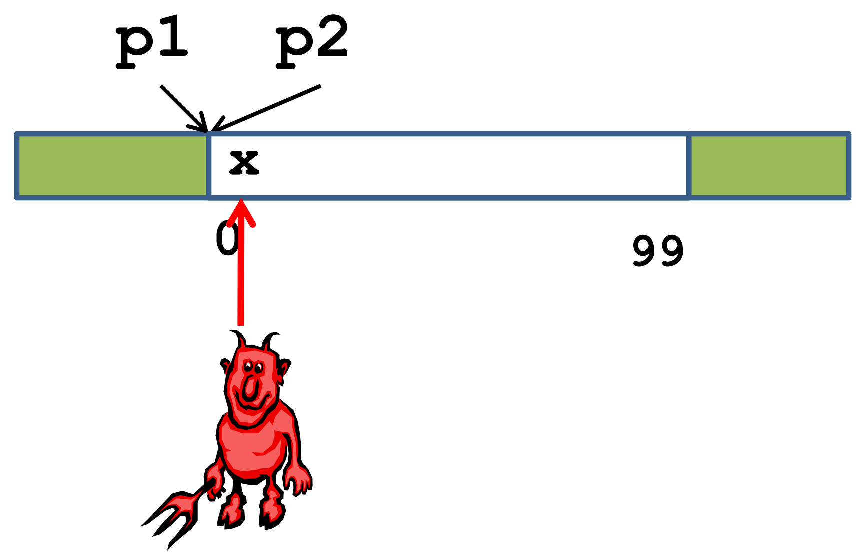
- Buffer overflow

```
char *c = malloc(100);
c[101] = 'a';
```



- Use after free

```
char *p1 = malloc(100);
char *p2 = p1;

free(p1);
p2[0] = 'x';
```



Strategies for Avoiding Memory Corruptions

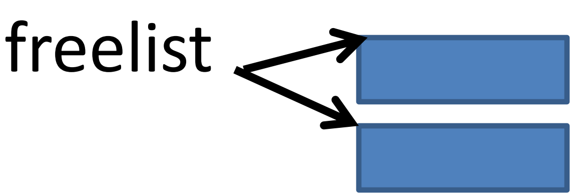
- Rewrite in a safe language (Java, C#, JavaScript)
- Static analysis / safe subset of C or C++
  - SAFECode [Adve], etc.
- Runtime detection, fail fast
  - Jones & Lin, CRED [Lam], CCured [Necula], others…

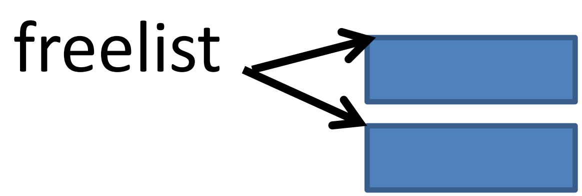
- A New Approach: Tolerate Corruption and Continue
  - Failure oblivious computing [Rinard] (unsound)
  - Rx, Boundless Memory Blocks, ECC memory
  - **DieHard / Exterminator, Samurai**

Correctness at What Cost?

- Heap implementations are/were maximally brittle for performance
- Space: packed as tightly as possible

- Time: reuse freed objects as soon as possible
  - free = push   malloc = pop

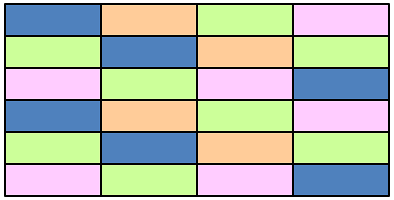




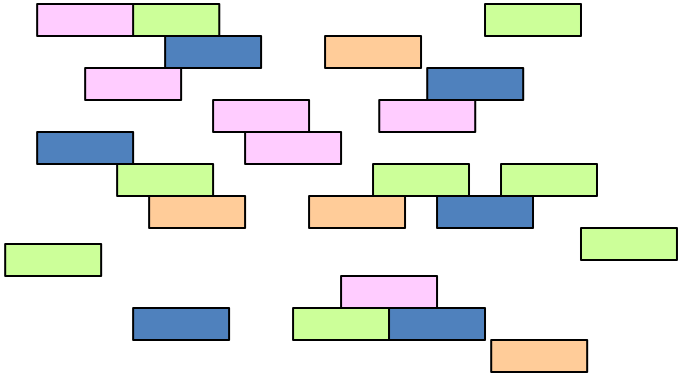
DieHard Allocator in a Nutshell

- With Emery Berger (PLDI 2006)
- Existing heaps are brittle, predictable
  - Predictable layout is easier for attacker to exploit
- Randomize and overprovision the heap
  - Expansion factor determines how much empty space
  - Semantics are identical
  - Allocator is easy to replace
- Replication increases benefits
- Exterminator extended ideas (PLDI 2007, Novark et al.)

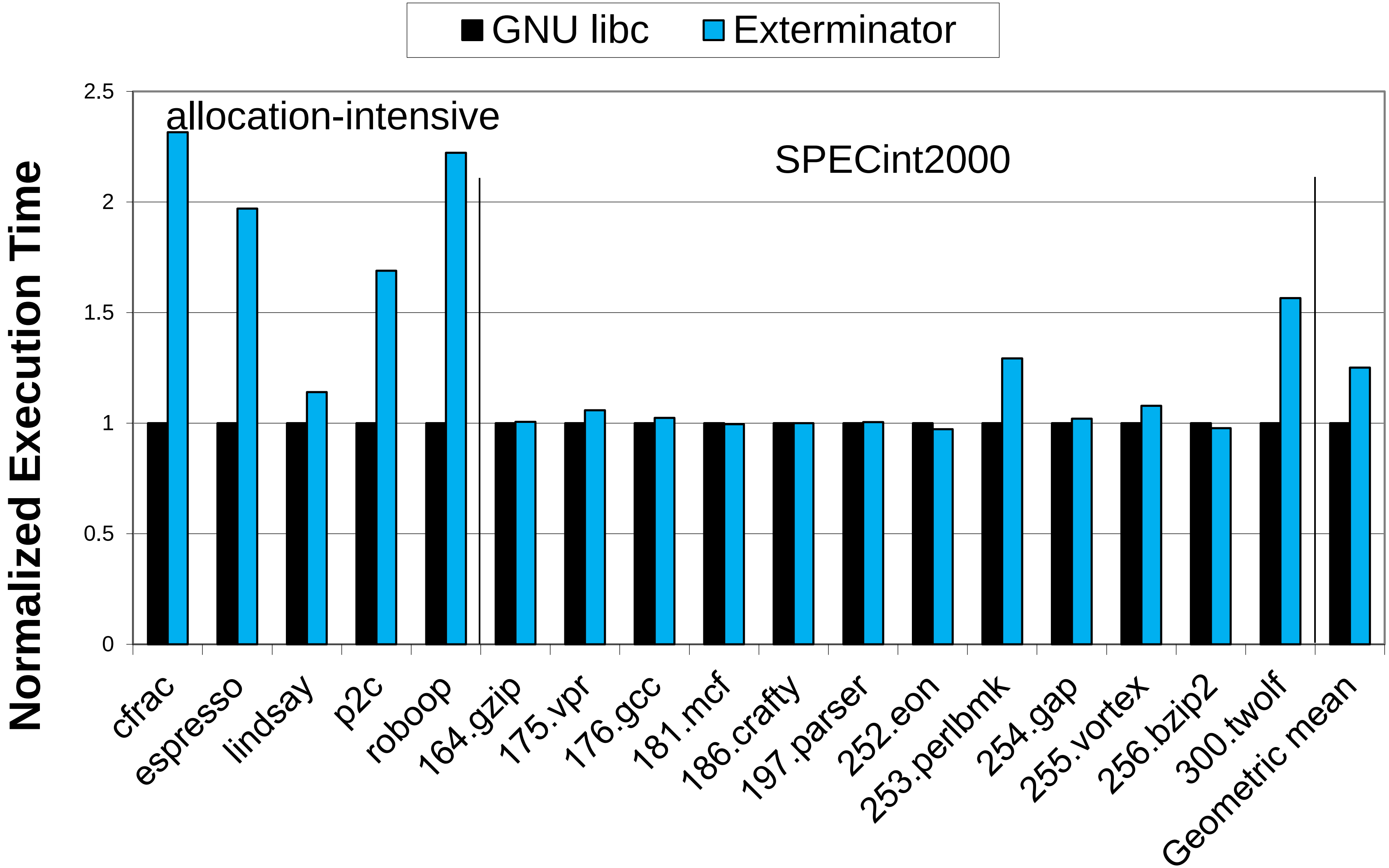
Normal Heap



DieHard Heap

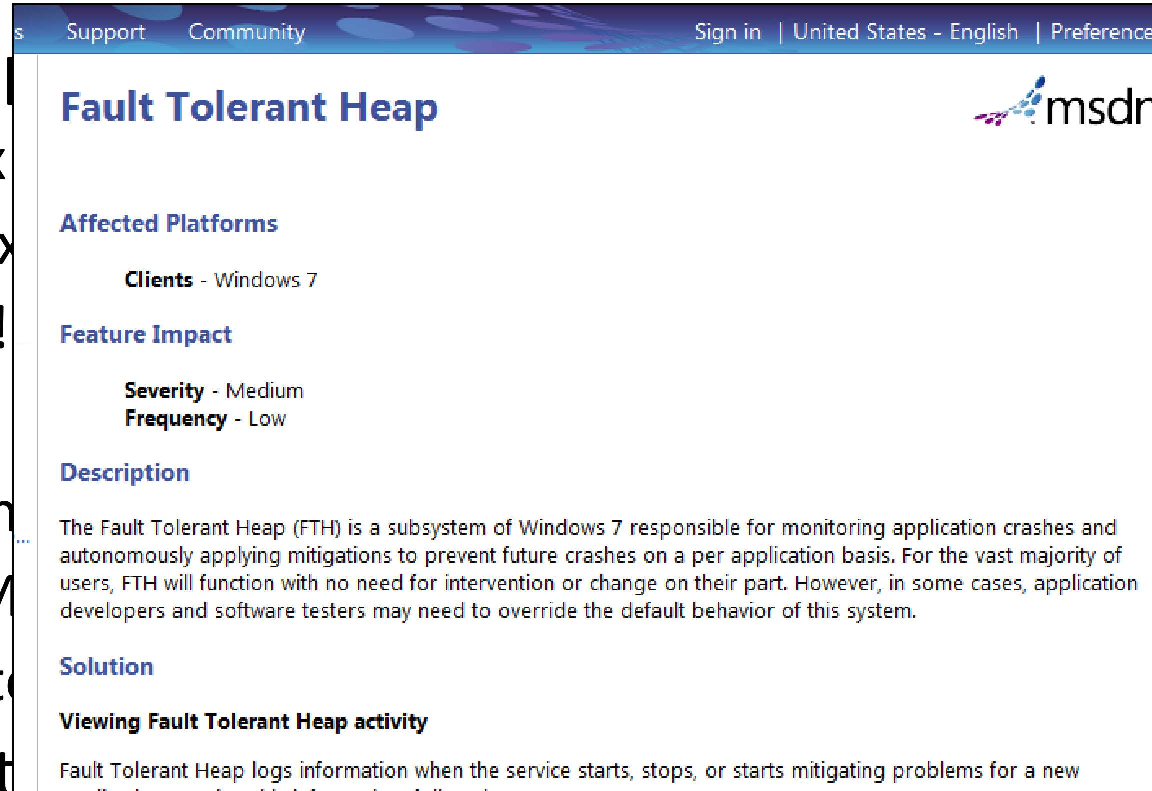


Of Course, Performance Matters



DieHard Impact

- DieHard (non-re
  - Windows, Linux
  - Works in FireFo
  - Try it right now!
- RobustHeap
  - Microsoft intern
  - Prototyped in M
  - Demonstrated t
- Windows 7 Fault
  - Inspired by ideas from DieHard/Robustheap
  - Turns on when application crashes



s Support Community Sign in | United States - English | Preferences

Fault Tolerant Heap

msdn

Affected Platforms

Clients - Windows 7

Feature Impact

Severity - Medium
Frequency - Low

Description

The Fault Tolerant Heap (FTH) is a subsystem of Windows 7 responsible for monitoring application crashes and autonomously applying mitigations to prevent future crashes on a per application basis. For the vast majority of users, FTH will function with no need for intervention or change on their part. However, in some cases, application developers and software testers may need to override the default behavior of this system.

Solution

Viewing Fault Tolerant Heap activity

Fault Tolerant Heap logs information when the service starts, stops, or starts mitigating problems for a new

A Benefit of Working at Microsoft…

One day I was trying to convince a security team that DieHard would improve security…

They said "What about heap spraying?"

And I said "What's that?" (long pause)

And they said "Look it up…"

Here's What I Found…


Thursday

When PDFs

The Shadowserve
Acrobat affecting
exploited. We are
sample last week
clear that we did
others are aware
Reader 8.1.0, 8.1.
confirmed via tes
will also affect it

…

However, it woul
you DISABLE JAV
functionality and
should be an easy

Disabling JavaScri
Click: Edit -> Pref

RITY WEBLOG

html

GOOGLE SEARCH

Custom Search

Search

BLOG ARCHIVE

▼ 2009 (140)

► August (11)

▼ July (33)

LuckySploit **
*.8866.org,podzone.o
ulaiba.net...

Spam ** 29 July

LuckySploit ** siyou.org

Spam ** 27July

LuckySploit **

FireEye Malware Intelligence Lab

Threat research, analysis, and mitigation | www.fireeye.com

Home | Archives | Subscribe

Background Summary

Most of the Acrobat exploits over the last several months use the, now common, heap spraying technique, implemented in Javascript/ECMAscript, a Turing complete language that Adobe thought would go well with static documents. (Cause that went so well for Postscript) (Ironically, PDF has now come full circle back to having the features of Postscript that it was trying to get away from.) The exploit could be made far far *less* reliable, by disabling Javascript in your Adobe Acrobat Reader.

But apparently there's no easy way to disable Flash through the UI. US-CERT recommends renaming the %ProgramFiles%\Adobe\Reader 9.0\Reader\authplay.dll and %ProgramFiles%\Adobe\Reader 9.0\Reader\rt3d.dll files. [Edit: Actually the source for this advice is the Adobe Product Security Incident Response Team (PSIRT).]

Anyway, here's why... Flash has it's own version of ECMAScript called Actionscript, and whoever wrote this new 0-day, finally did something new by implementing the heap-spray routine with Actionscript inside of Flash.

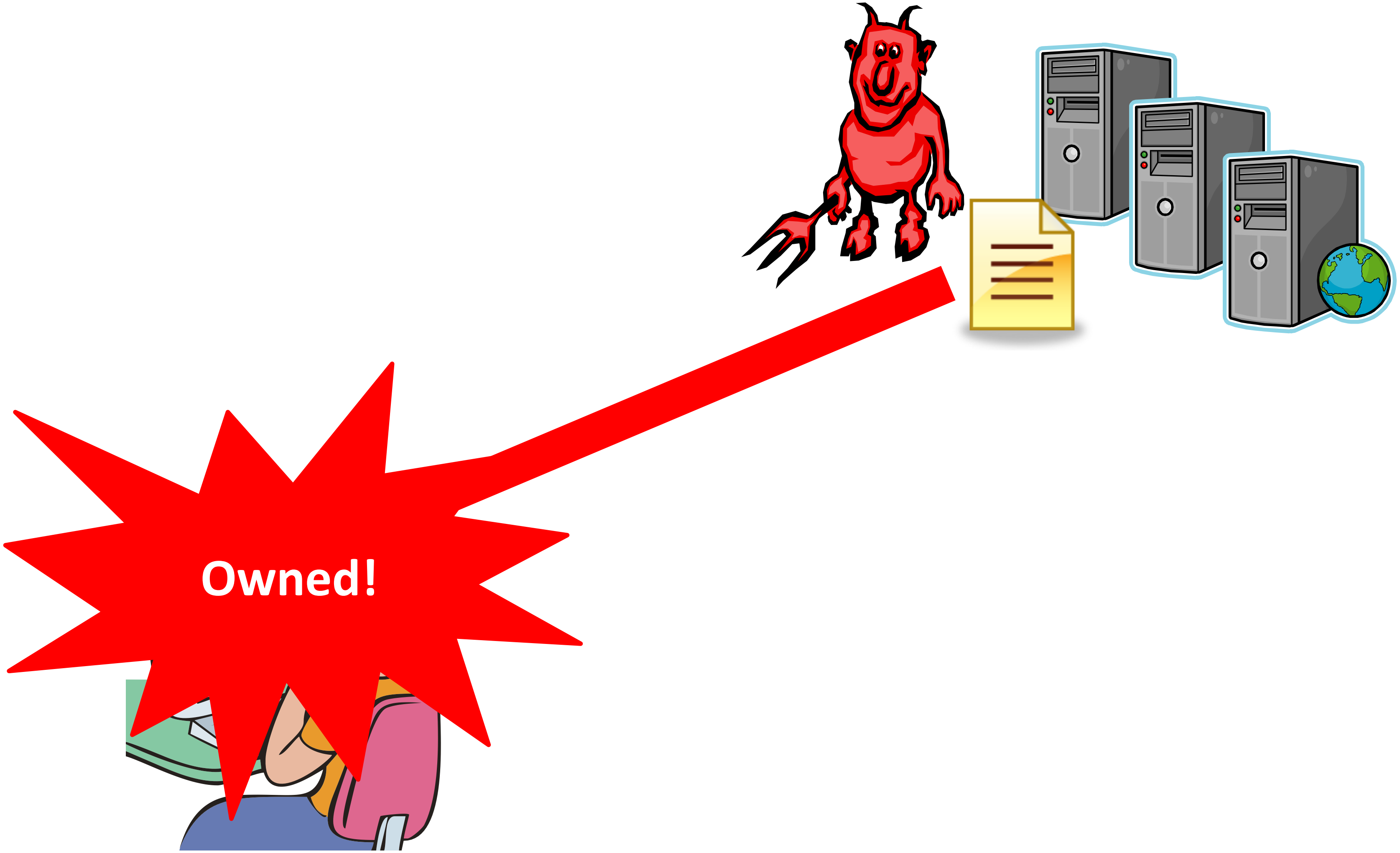
Details

Common Element:

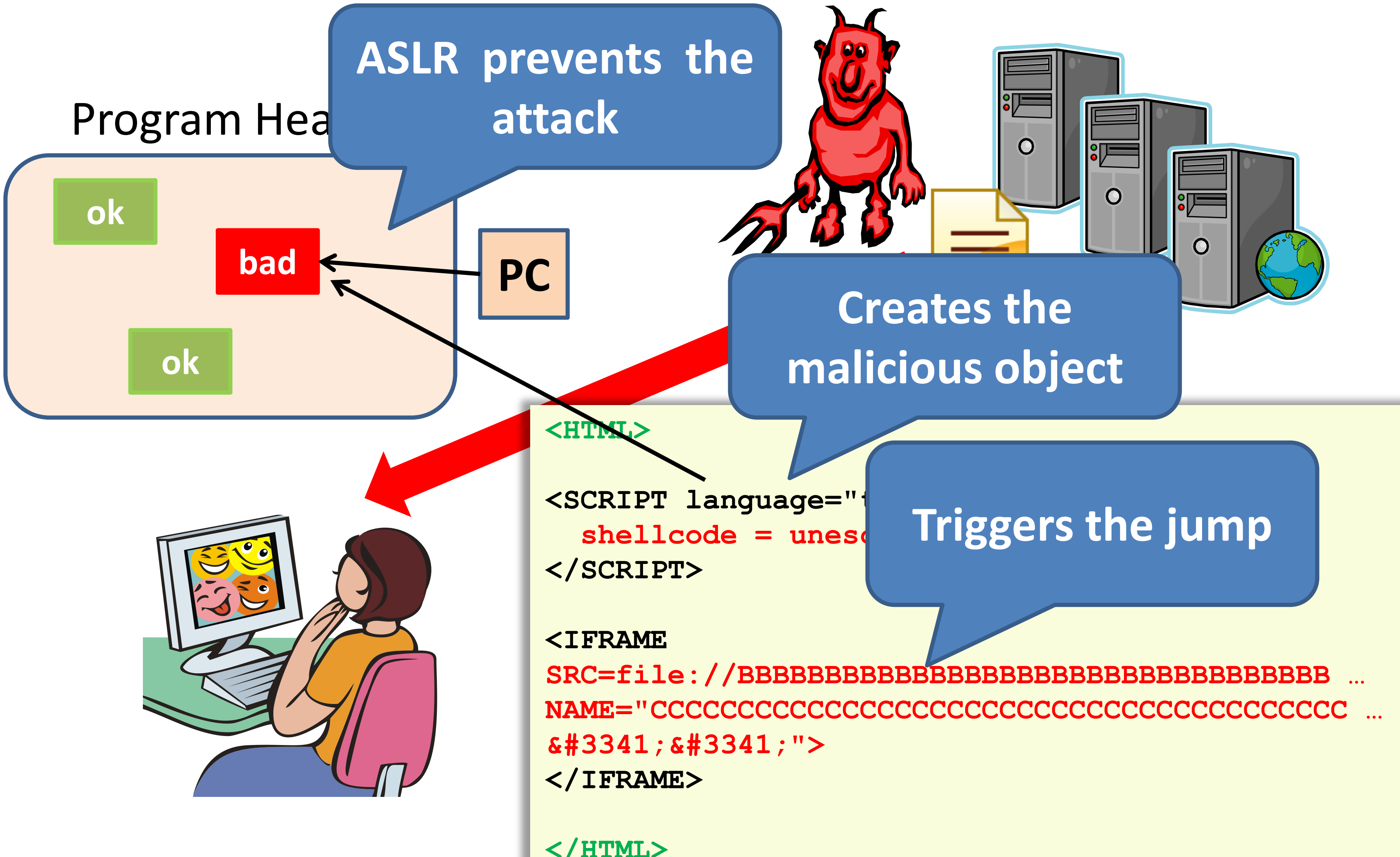
All vulnerable applications support embedded scripting languages (JavaScript, ActionScript, etc.)

http://blog.fireeye.com/research/2009/07/actionscript_heap_spray.html

Drive-By Heap Spraying



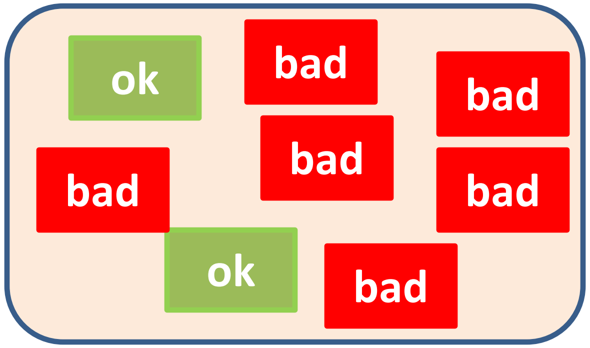
Drive-By Heap Spraying (2)

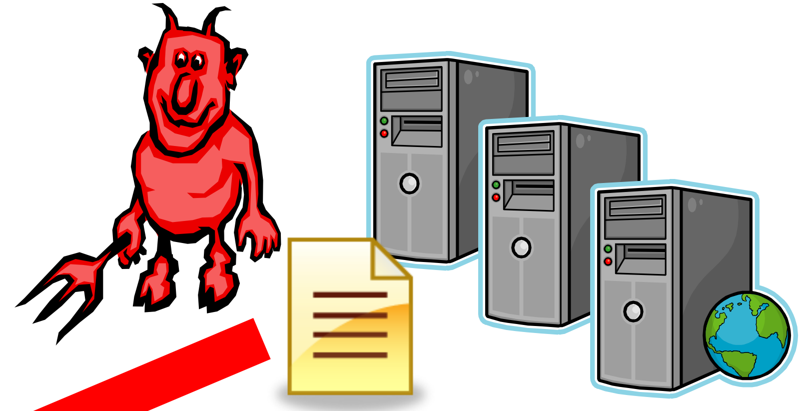


RiSE

Drive-By Heap Spraying (3)

Program Heap





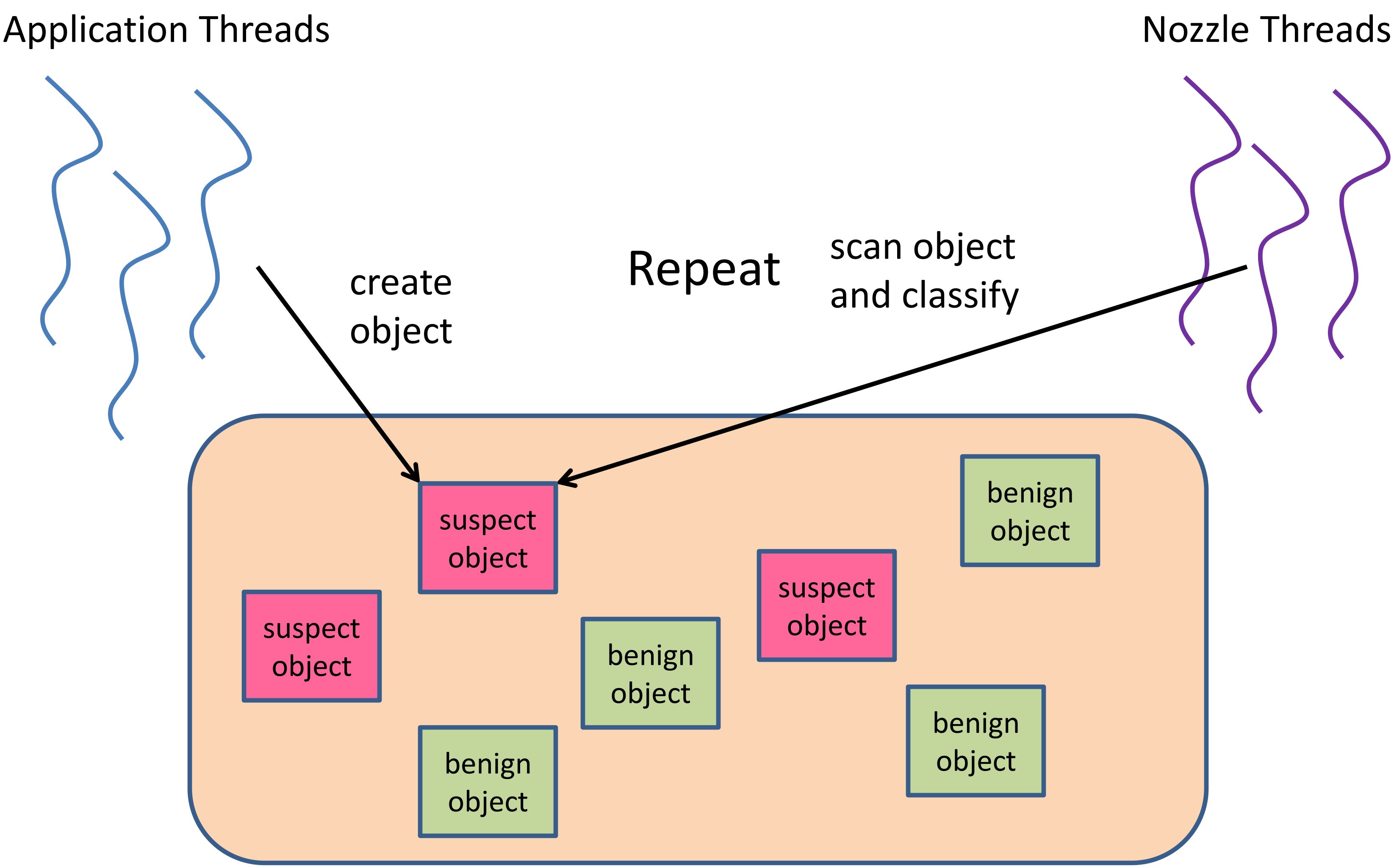
```
<SCRIPT language="text/javascript">
  shellcode = unescape("%u4343%u4343%...'');
  oneblock = unescap
  var fullblock = on
  while (fullblock.l
    fullblock += ful
  }

  sprayContainer = new Array();
  for (i=0; i<1000; i++) {
    sprayContainer[i] = fullblock + shellcode;
  }
</SCRIPT>
```

Allocate 1000s of malicious objects

Nozzle – Detecting Heap Spraying

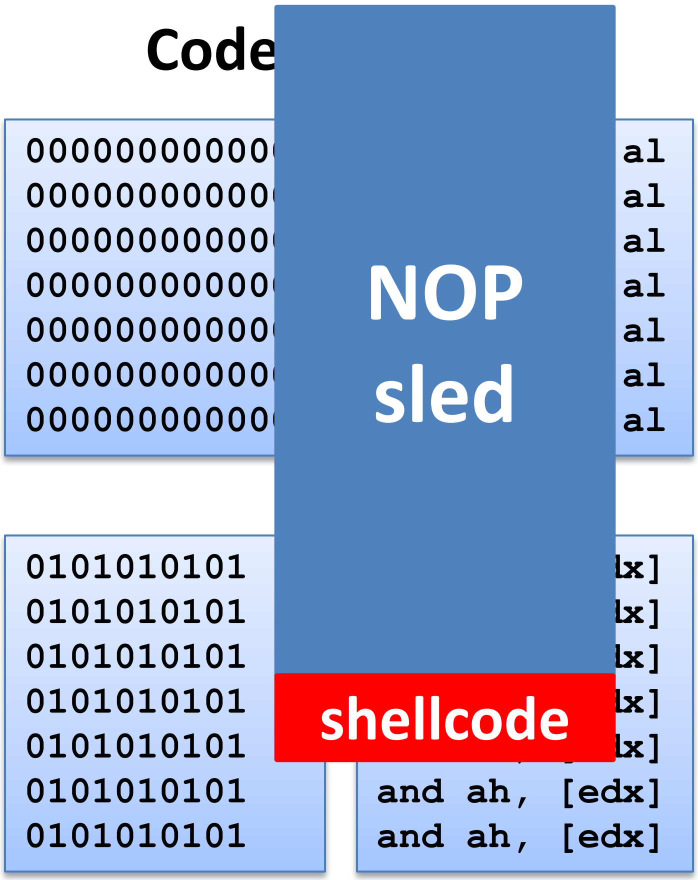
- Joint work with Paruj Ratanaworabhan (Kasetsart University) and Ben Livshits (Microsoft Research)
- Insight:
  - Spraying creates many objects with malicious content
  - That gives the heap unique, recognizable characteristics
- Approach:
  - Dynamically scan objects to estimate overall malicious content

Nozzle: Classifying Malicious Objects



Local Malicious Object Detection

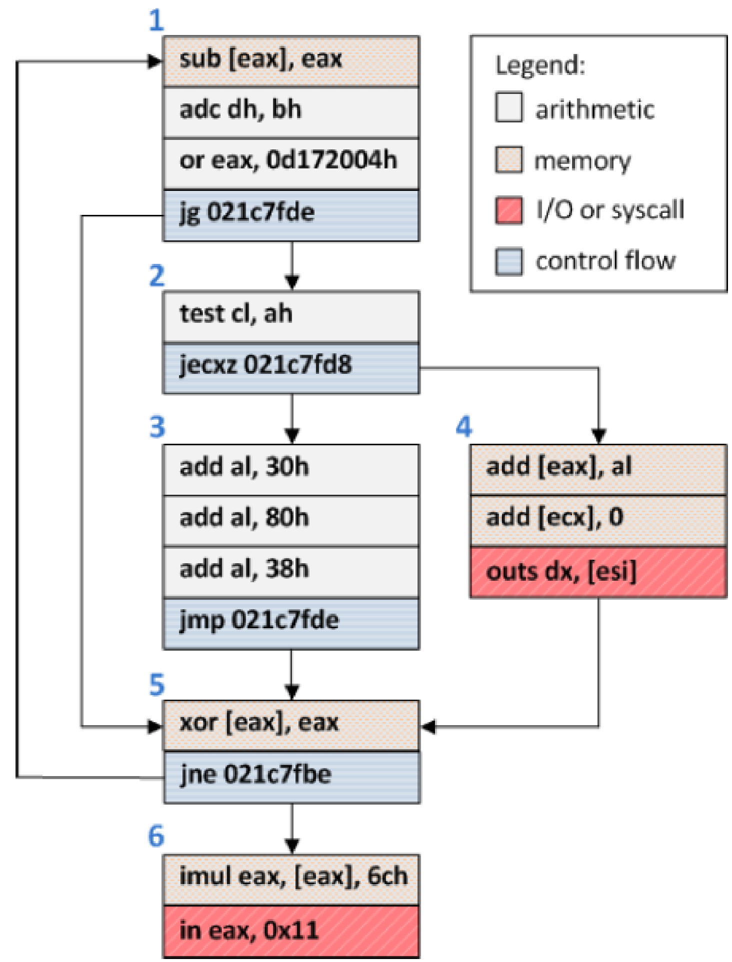
Is this object dangerous?



- Is this object code?
  - Code and data look the same on x86
- Focus on sled detection
  - Majority of object is sled
  - Spraying scripts build simple sleds
- Is this code a NOP sled?
  - Previous techniques do not look at heap
  - Many heap objects look like NOP sleds
  - 80% false positive rates using previous techniques
- Need stronger local techniques

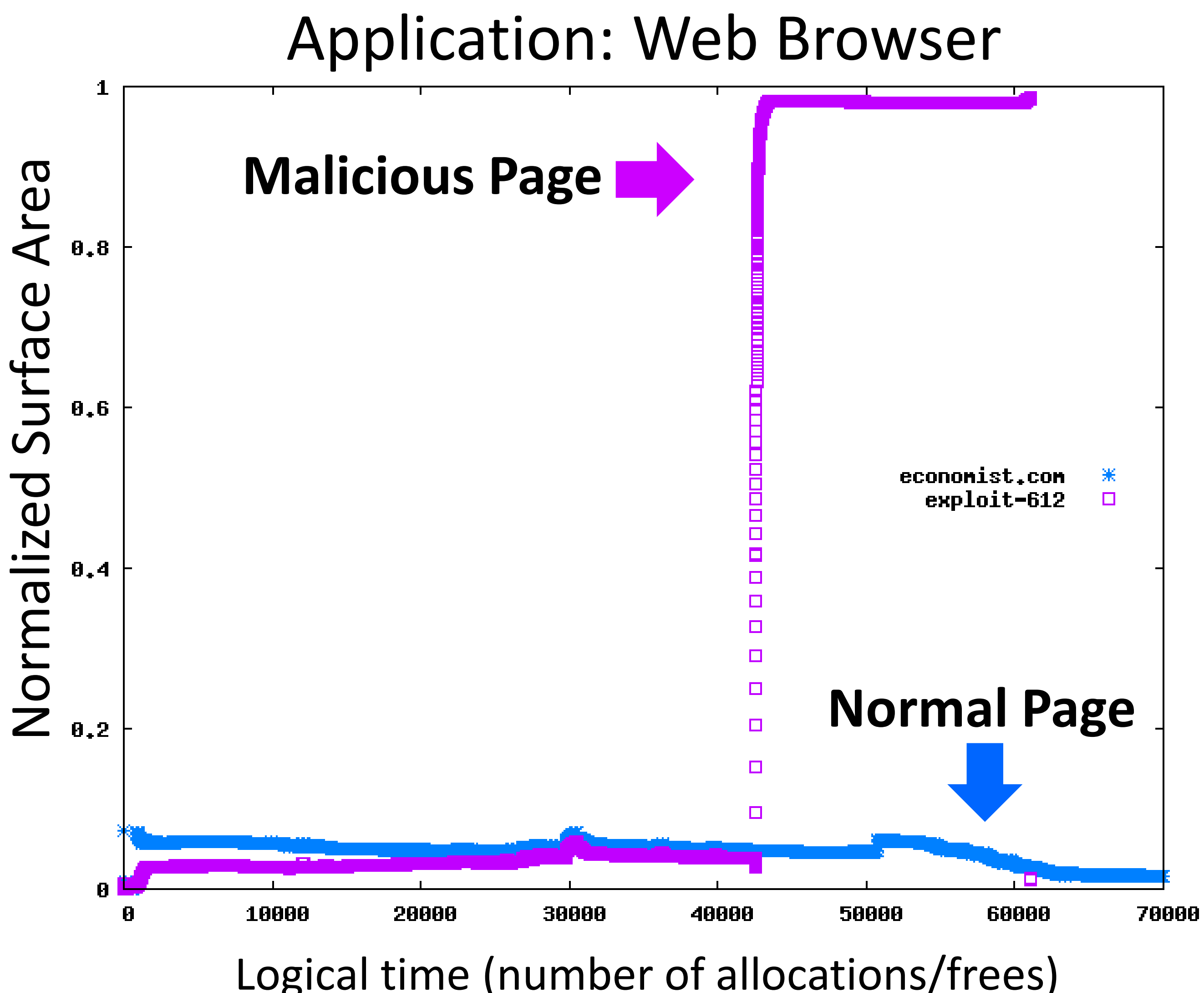
Object Surface Area Calculation

- Assume: attacker wants to reach shell code from jump to any point in object
- Goal: find blocks that are likely to be reached via control flow
- Strategy: use dataflow analysis to compute "surface area" of each block

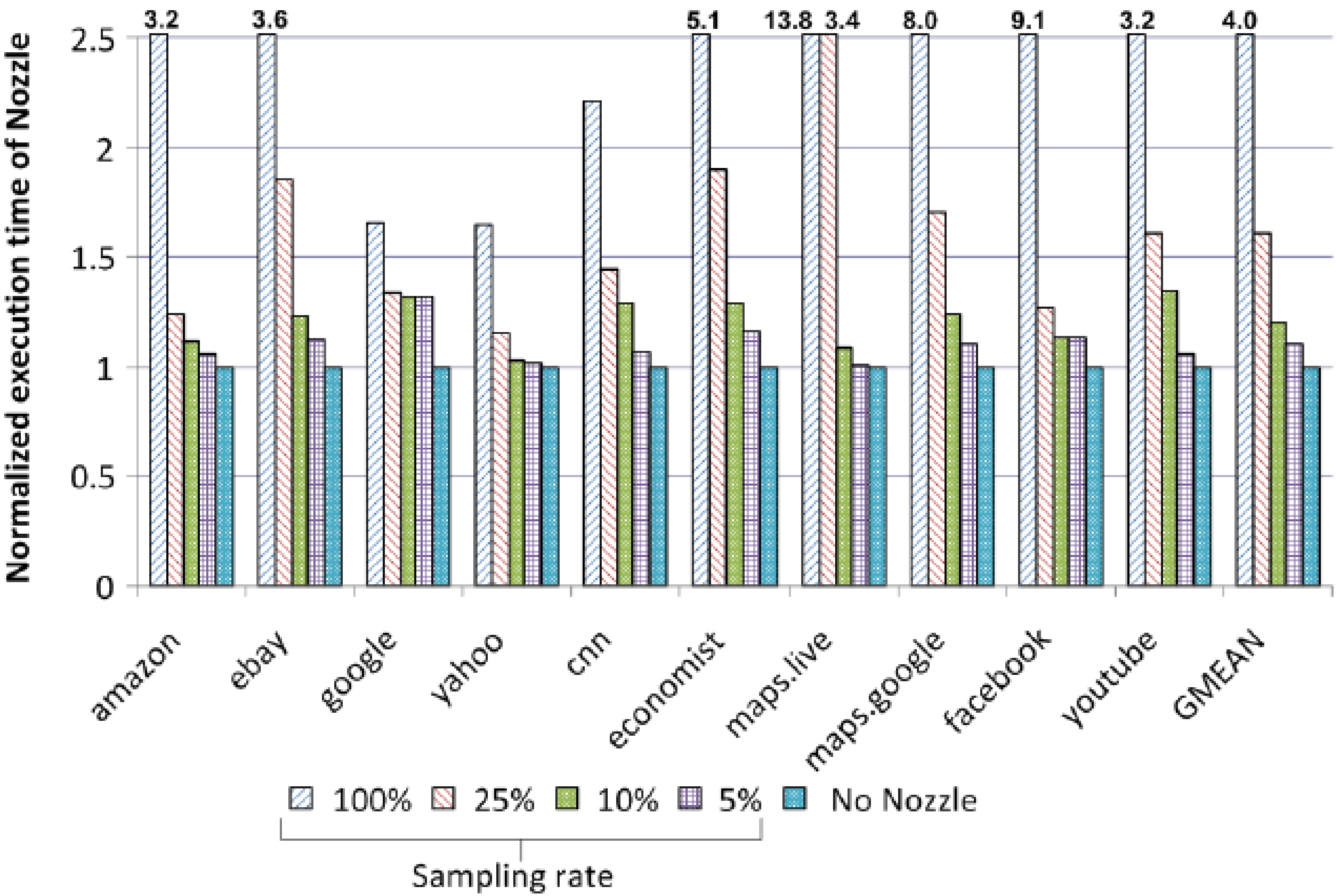


An example object from visiting google.com

Nozzle Effectiveness

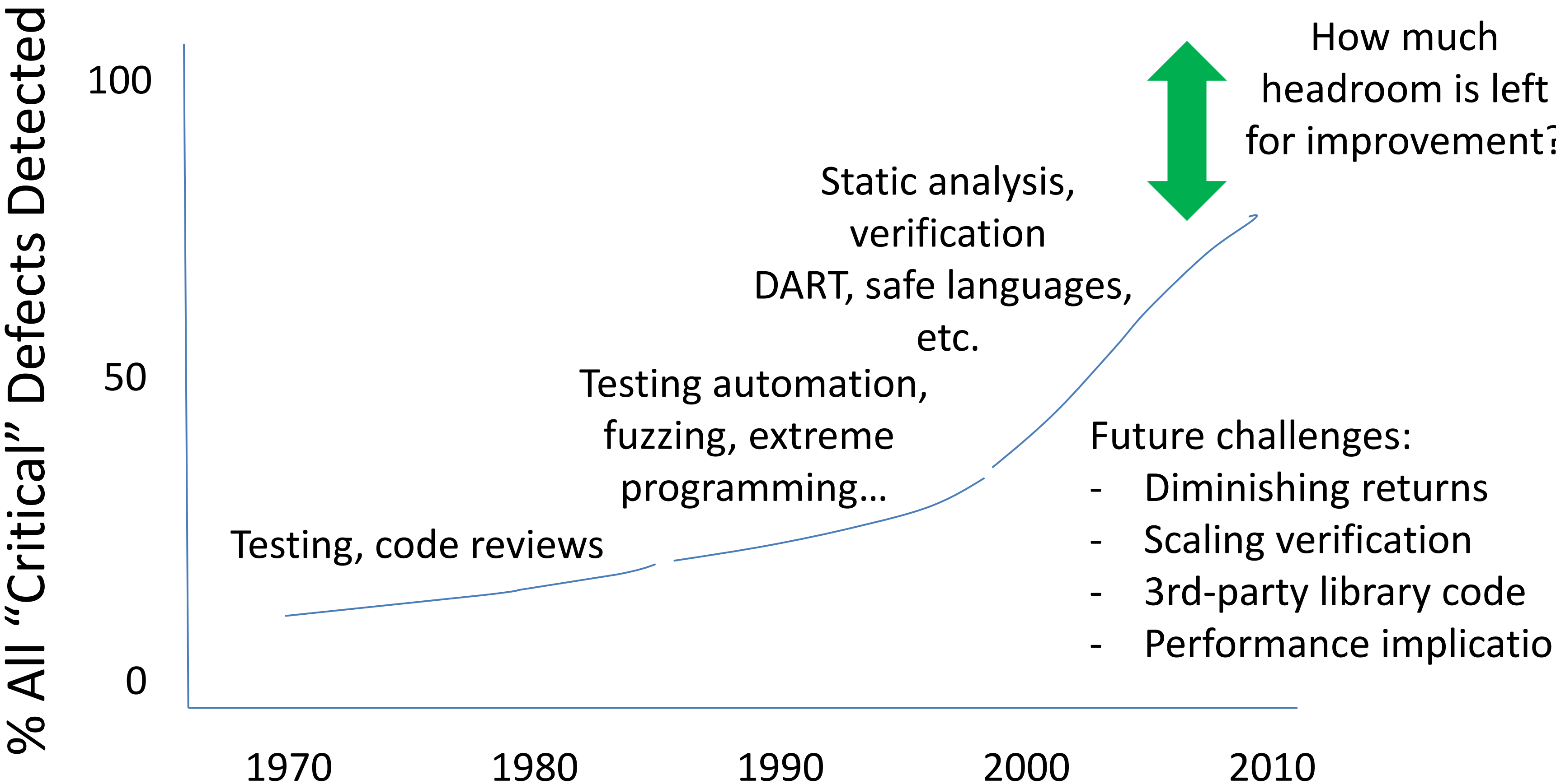


Nozzle Performance



So, Performance is Dead…

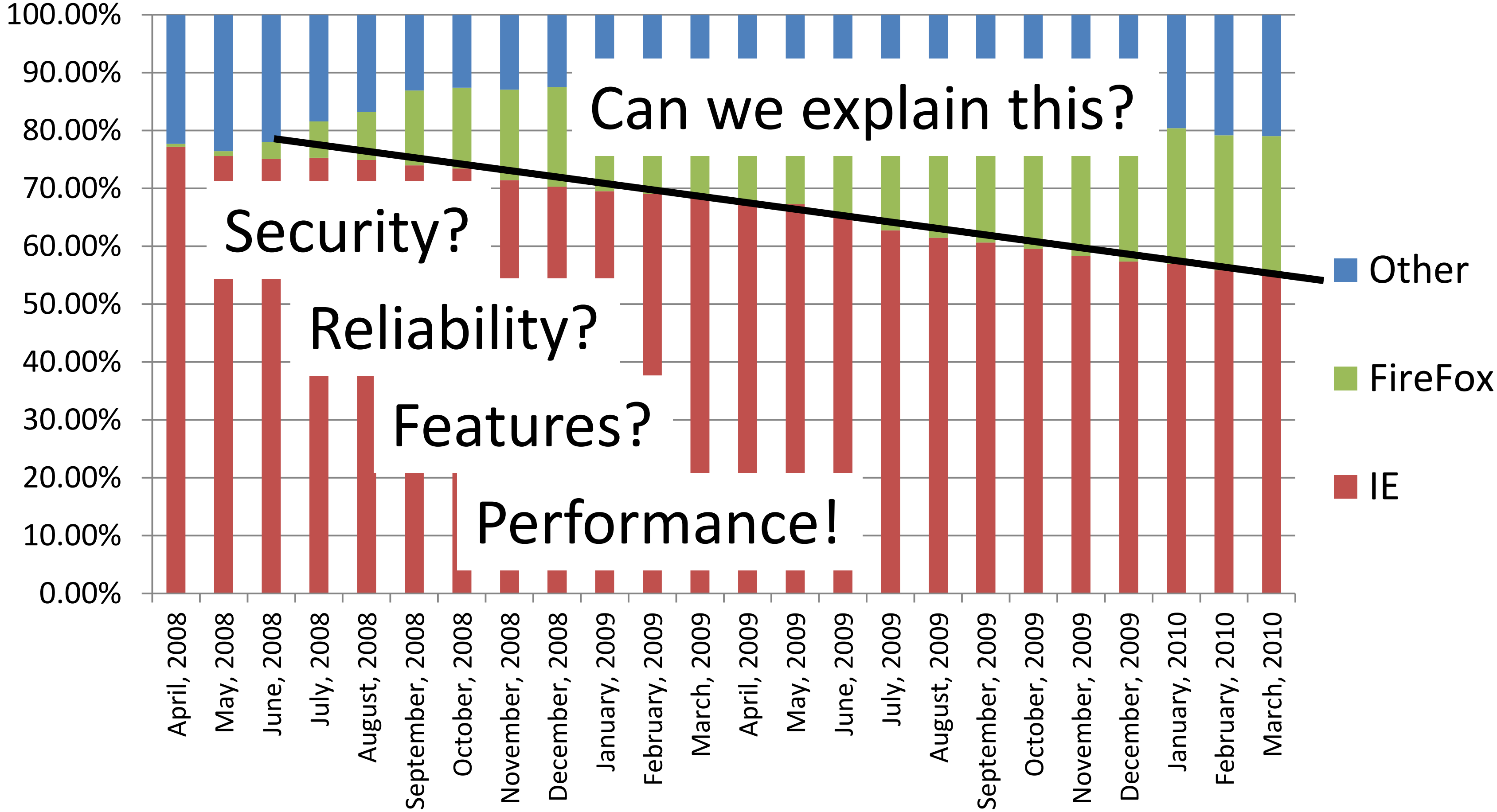
How far can defect detection and runtime toleration go?



Future challenges:

- Diminishing returns
- Scaling verification
- 3rd-party library code
- Performance implications

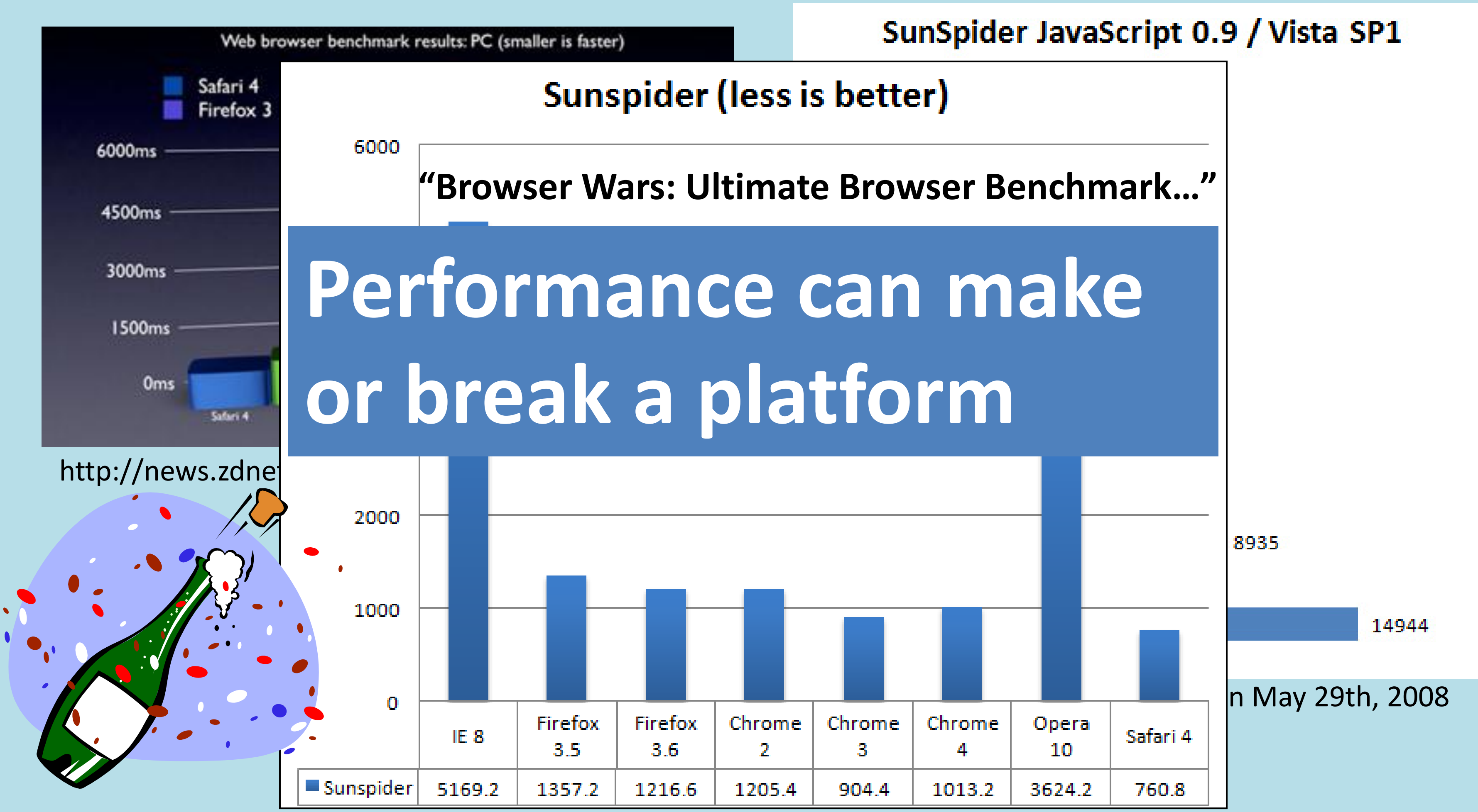
What's Happening Here? Browser Market Share Trends



Source: http://marketshare.hitslink.com/

Long Live Performance!

"Safari dominates browser benchmarks" **"Browser faceoff: IE vs Firefox vs Opera vs Safari"**



"Browser Wars: Ultimate Browser Benchmark…"

Performance can make or break a platform

http://news.zdne…

…n May 29th, 2008

http://www.favbrowser.com/chrome-vs-opera-vs-firefox-vs-internet-explorer-vs-safari/

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

Understanding JavaScript Behavior

With Paruj Ratanaworabhan and Ben Livshits

Benchmarks

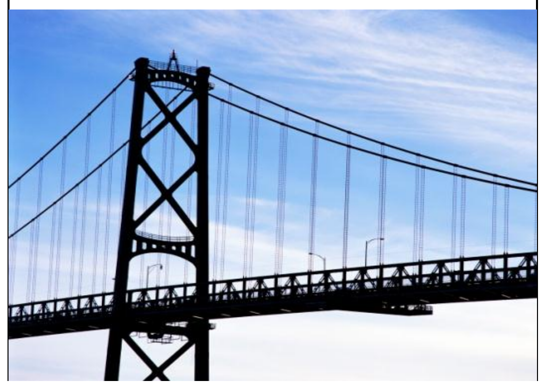
7 V8 programs:

- richards
- deltablue
- crypto
- raytrace
- earley-boyer
- regexp
- splay

8 SunSpider programs:

- 3-draytrace
- access-nbody
- bitops-nsieve
- controlflow
- crypto-aes
- date-xparb
- math-cordic
- string-tagcloud

JSMeter



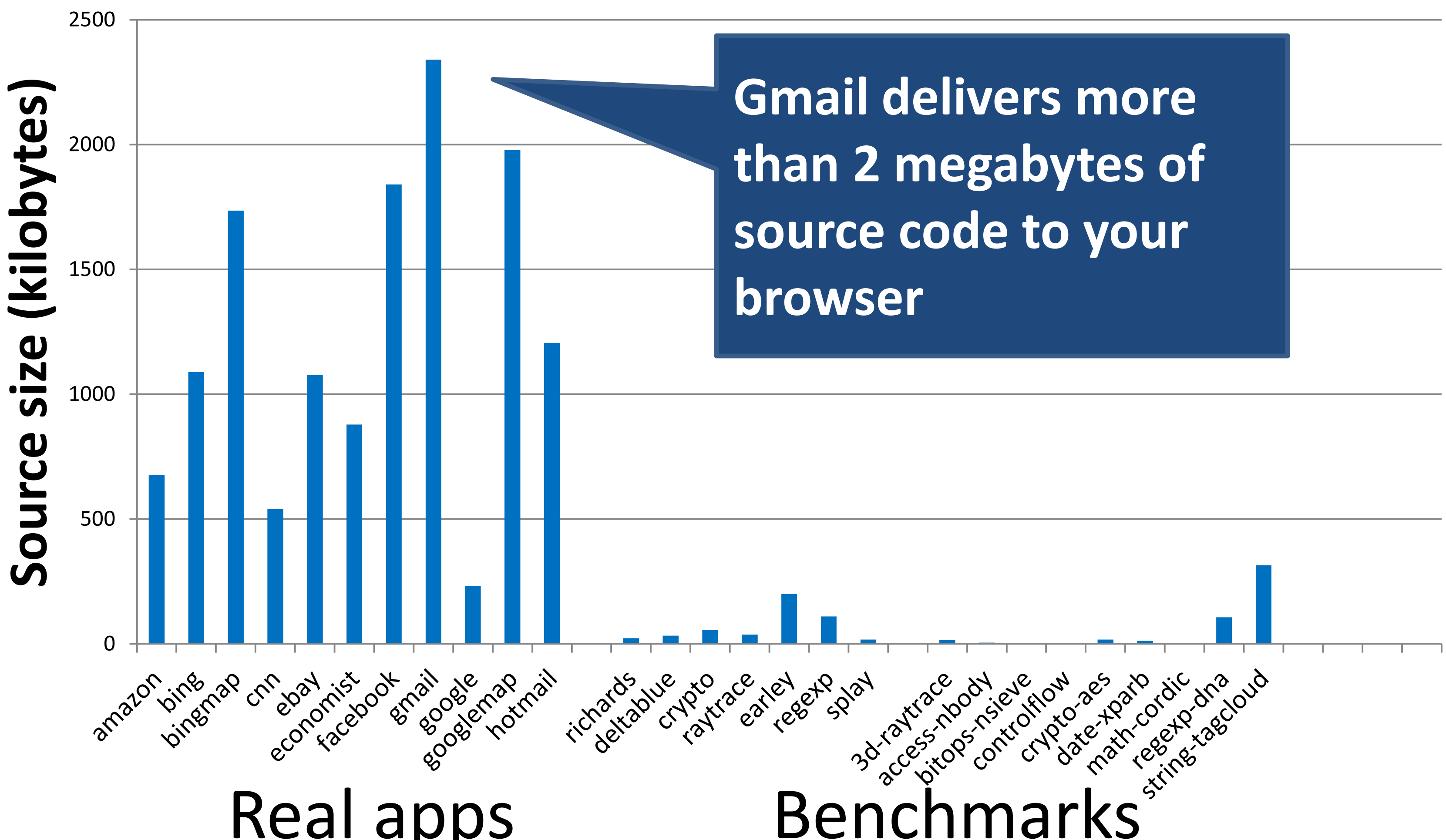
Real apps

bing, Google, Gmail, Windows Live Hotmail, amazon.com, ebay, bing Maps, Google Maps, The Economist, CNN, facebook

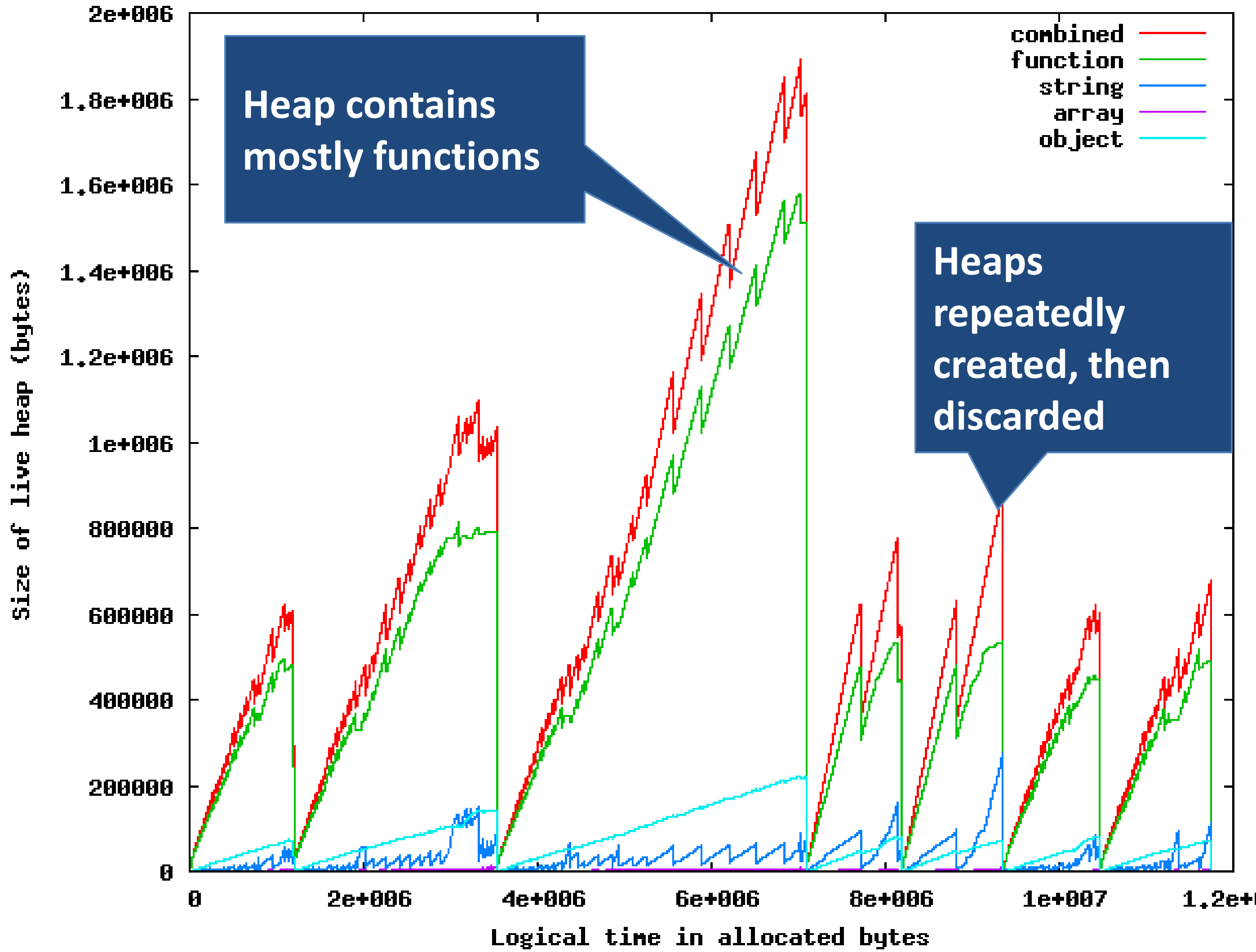
Goal: Measure JavaScript in real web applications

Approach: Instrument IE runtime

Real Apps are Much Bigger

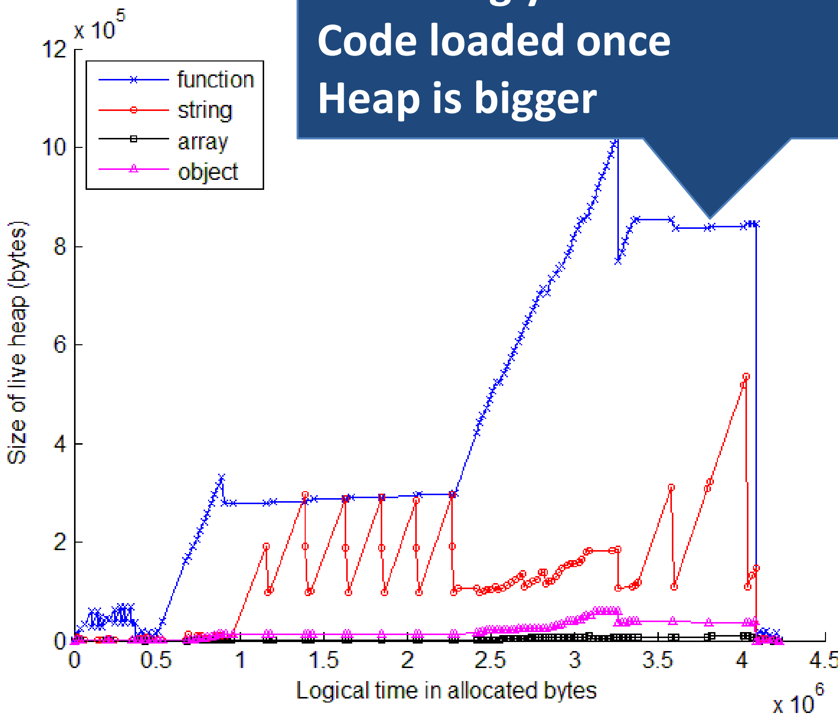


Real Apps have Interesting Behavior: Live Heap over Time (eBay)



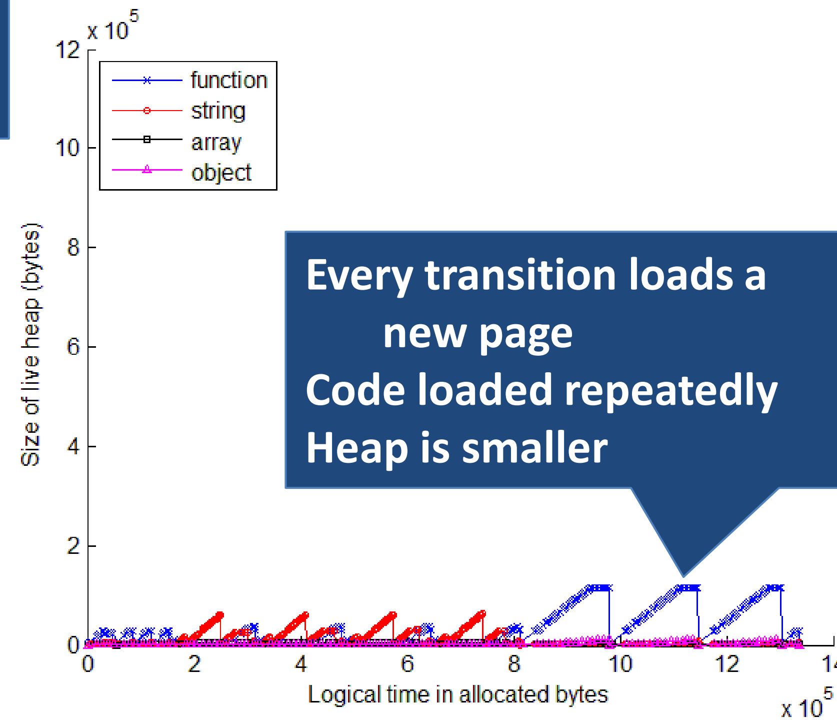
Real Apps have Different Architectures

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)

The Next 10 Years

- Reliability
- “Good enough” = cheap
- Energy

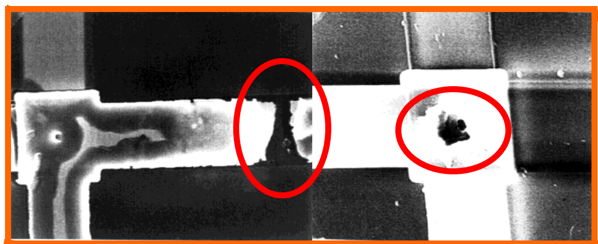
- Concurrency

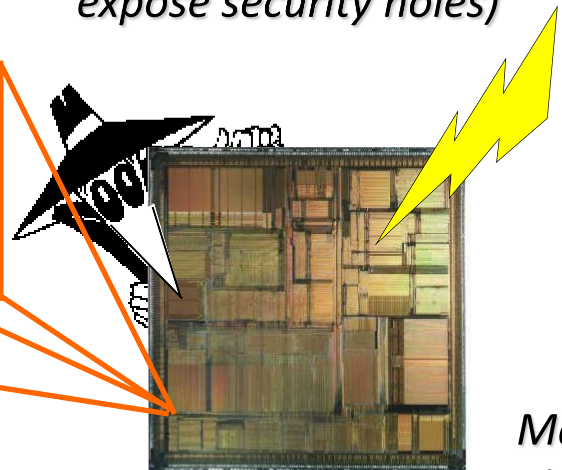
Reliability Threats

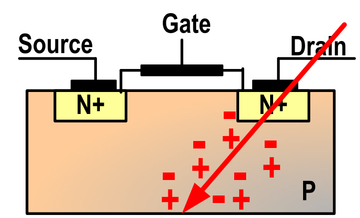
Silicon Defects
(Manufacturing defects and device wear-out)

H/W and S/W Design Errors
(Bugs are expensive and expose security holes)

Transient Faults due to Cosmic Rays & Alpha Particles
(Increase exponentially with number of devices on chip)



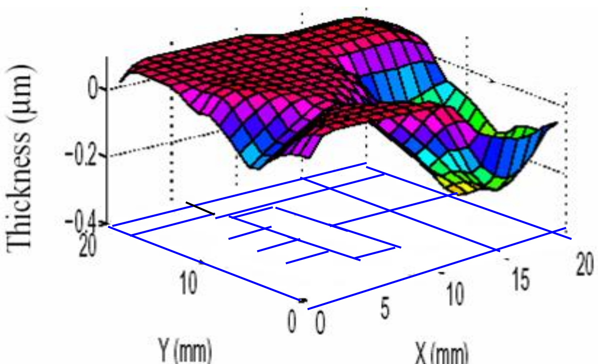


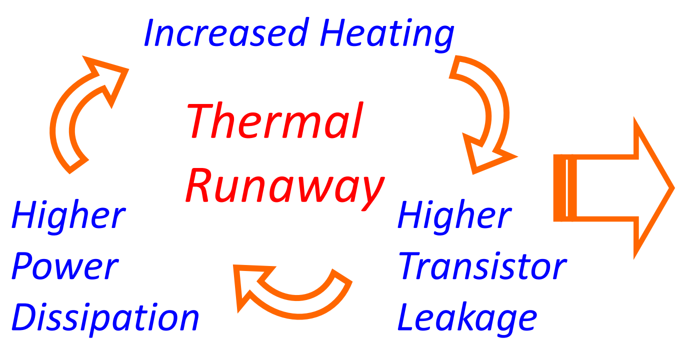


Parametric Variability
(Uncertainty in device and environment)

Manufacturing Defects That Escape Testing
(Inefficient Burn-in Testing)

Intra-die variations in ILD thickness





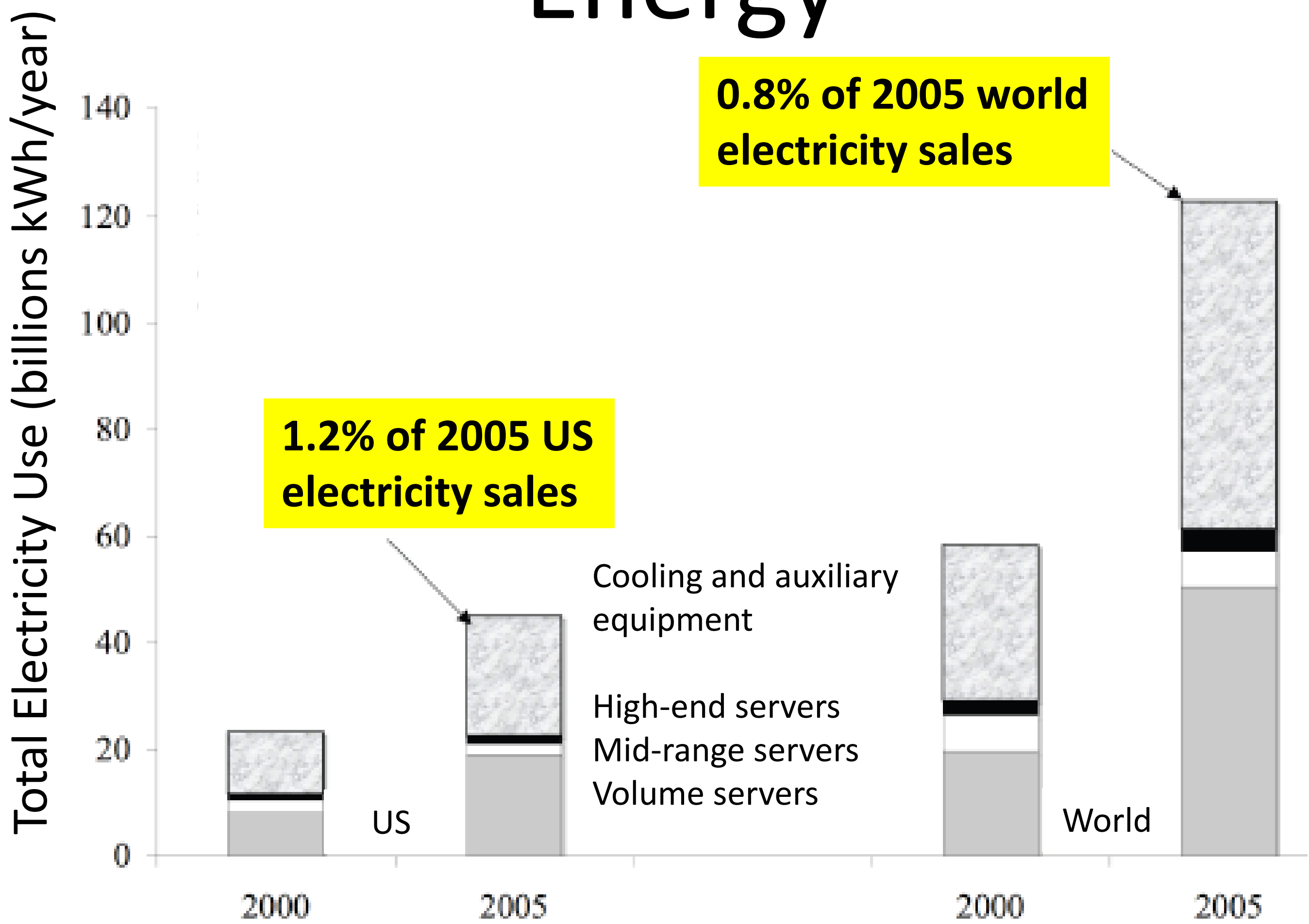
Slide courtesy of Todd Austin “Reliable Processor Research @ Umich”

The "Good Enough" Revolution

Source: WIRED Magazine (Sep 2009) – Robert Kapps

- Observation: People prefer "cheap and good enough" over "costly and near-perfect"
- Examples: Flip video cameras, Skype, etc.
- Conclusion:
  - Engineer for imperfect result at low cost
- Projects: Green (Chilimbi, MSR), Perforation (Rindard, MIT), Flicker (Pattabiraman, UBC)

Energy



"Estimating Total Power Consumption by Servers in the U.S. and the World", Jonathan G. Koomey, LBL Report, Feb. 2007

Conclusions

- Performance was and continues to be critical
  - Correctness and security neglected until 2000s
- What is being optimized changes
  - Energy usage
  - Concurrency
  - Cost effectiveness
  - Constrained devices
- Improvements in next 10 years harder
  - Proebsting’s Law: Accurate? Acceptable?

Acknowledgements

- CGO Organizers (especially Kim Hazelwood and David Kaeli)
- Todd Austin, U. Michigan
- Alex David, Deborah Robinson – Microsoft
- Mark Horowitz, Ofer Shacham – Stanford
- CJ Newburn, Shubu Mukherjee – Intel
- Karthik Pattabiraman – UBC
- DBLP Computer Science Bibliography – Universität Trier

Questions?