Designing and implementing secure web browsers
or
How to keep your cores busy for two seconds at a time

Chris Grier, Shuo Tang, Samuel T. King

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Motivation

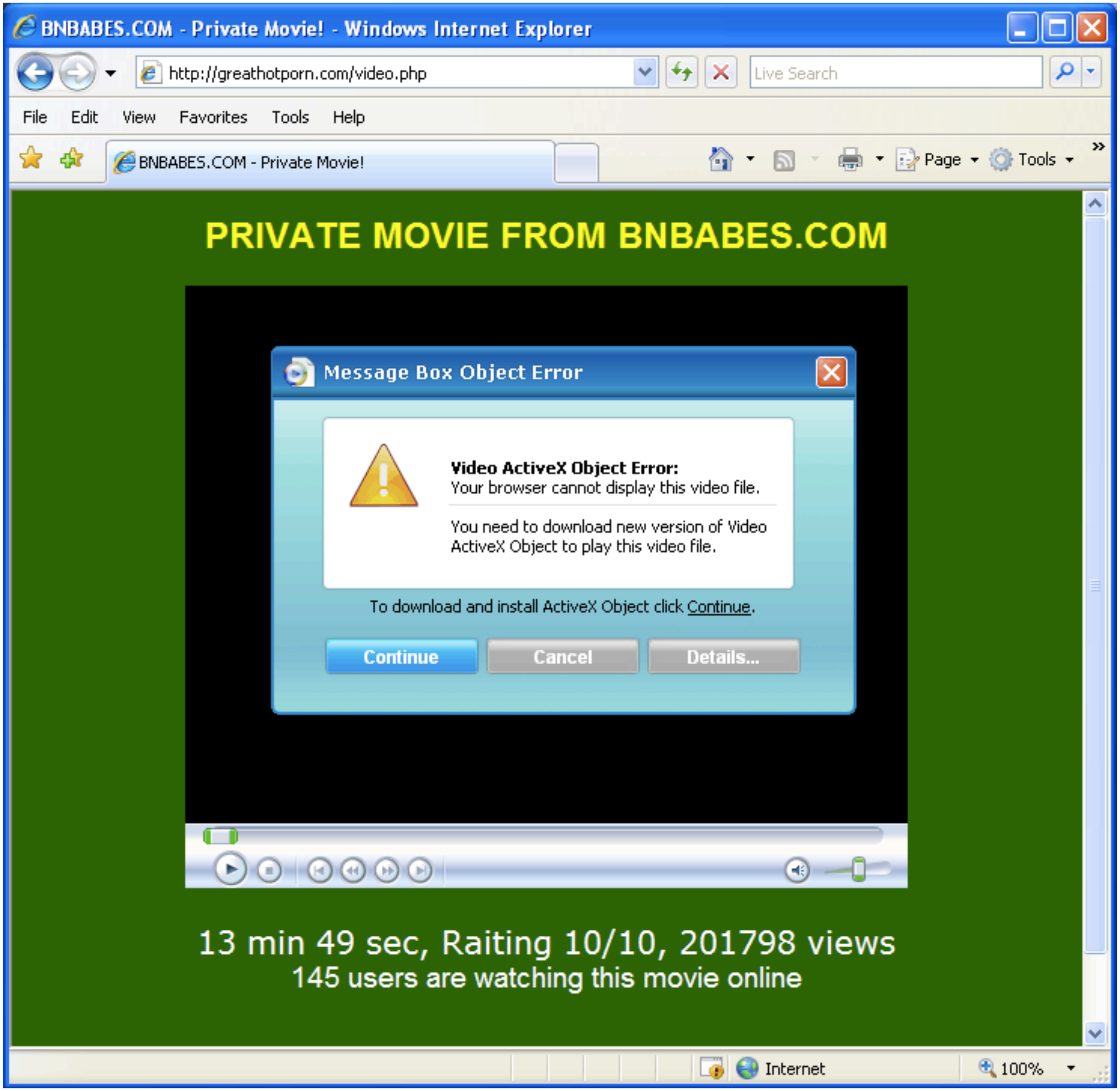
- Browser most commonly used application today
- Browsers are an application platform
  - Email, banking, investing, shopping, television, and more!

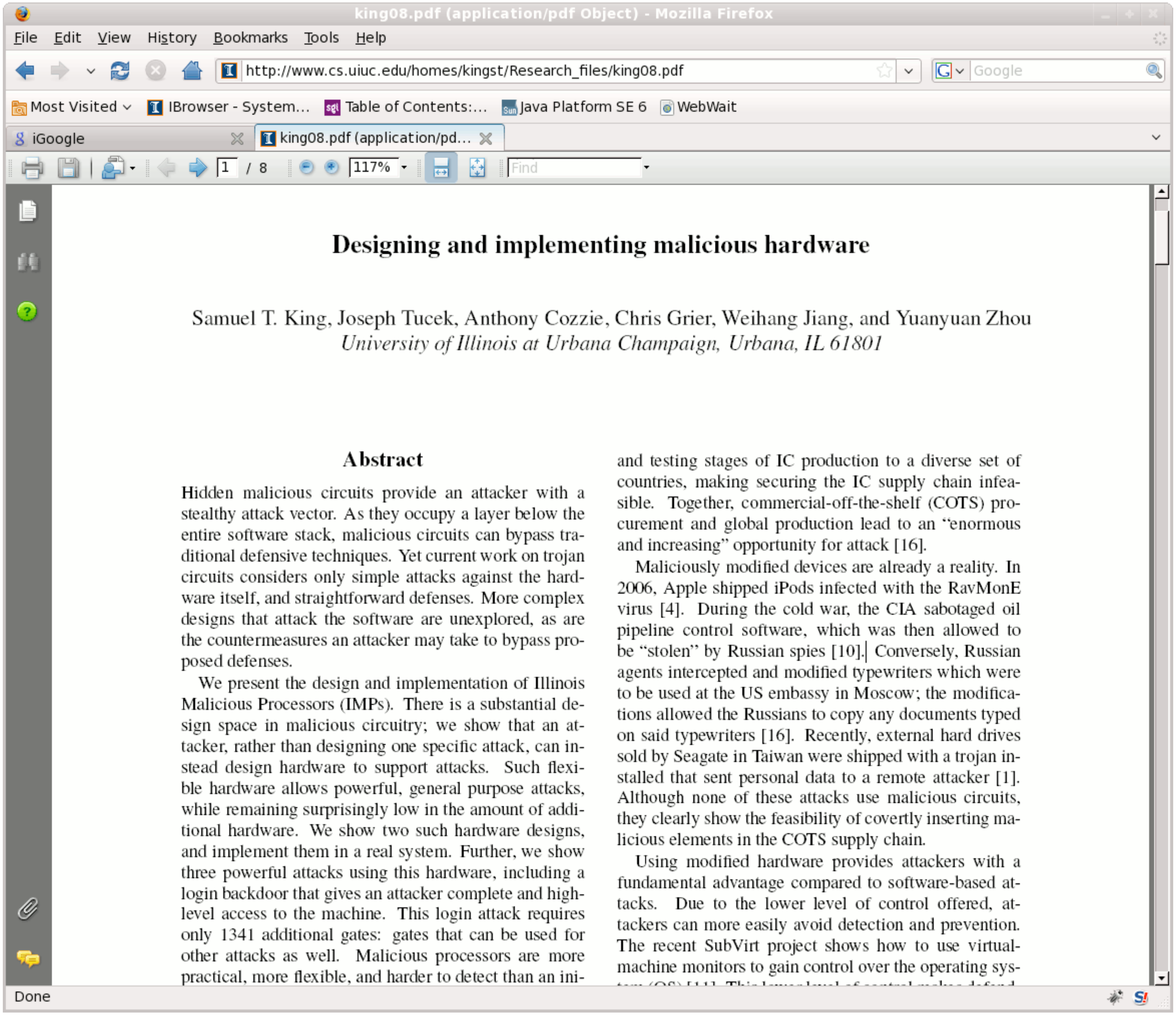
- Browsers are plagued with vulnerabilities
  - Internet Explorer: 57 vulnerabilities
  - Mozilla/Firefox: 122 vulnerabilities
  - Safari + Opera: 66 vulnerabilities

- Studies from Microsoft, Google, and University of Washington show web browser is attacker target

Anatomy of a browser attack

- What could a browser attack look like?


BNBABES.COM - Private Movie! - Windows Internet Explorer
http://greathotporn.com/video.php
Live Search
File Edit View Favorites Tools Help
BNBABES.COM - Private Movie!
Page Tools
PRIVATE MOVIE FROM BNBABES.COM
Message Box Object Error
Video ActiveX Object Error:
Your browser cannot display this video file.
You need to download new version of Video ActiveX Object to play this video file.
To download and install ActiveX Object click Continue.
Continue
Cancel
Details...
13 min 49 sec, Raiting 10/10, 201798 views
145 users are watching this movie online
Internet
100%



Designing and implementing malicious hardware

Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou
University of Illinois at Urbana Champaign, Urbana, IL 61801

Abstract

Hidden malicious circuits provide an attacker with a stealthy attack vector. As they occupy a layer below the entire software stack, malicious circuits can bypass traditional defensive techniques. Yet current work on trojan circuits considers only simple attacks against the hardware itself, and straightforward defenses. More complex designs that attack the software are unexplored, as are the countermeasures an attacker may take to bypass proposed defenses.

We present the design and implementation of Illinois Malicious Processors (IMPs). There is a substantial design space in malicious circuitry; we show that an attacker, rather than designing one specific attack, can instead design hardware to support attacks. Such flexible hardware allows powerful, general purpose attacks, while remaining surprisingly low in the amount of additional hardware. We show two such hardware designs, and implement them in a real system. Further, we show three powerful attacks using this hardware, including a login backdoor that gives an attacker complete and high-level access to the machine. This login attack requires only 1341 additional gates: gates that can be used for other attacks as well. Malicious processors are more practical, more flexible, and harder to detect than an ini-

and testing stages of IC production to a diverse set of countries, making securing the IC supply chain infeasible. Together, commercial-off-the-shelf (COTS) procurement and global production lead to an "enormous and increasing" opportunity for attack [16].

Maliciously modified devices are already a reality. In 2006, Apple shipped iPods infected with the RavMonE virus [4]. During the cold war, the CIA sabotaged oil pipeline control software, which was then allowed to be "stolen" by Russian spies [10]. Conversely, Russian agents intercepted and modified typewriters which were to be used at the US embassy in Moscow; the modifications allowed the Russians to copy any documents typed on said typewriters [16]. Recently, external hard drives sold by Seagate in Taiwan were shipped with a trojan installed that sent personal data to a remote attacker [1]. Although none of these attacks use malicious circuits, they clearly show the feasibility of covertly inserting malicious elements in the COTS supply chain.

Using modified hardware provides attackers with a fundamental advantage compared to software-based attacks. Due to the lower level of control offered, attackers can more easily avoid detection and prevention. The recent SubVirt project shows how to use virtual-machine monitors to gain control over the operating sys-


Yahoo! - Opus Palladianum
File Edit View History Bookmarks Window Tools Help
http://www.yahoo.com/
Google
UIUC CS Slashdot NYTimes ESPN Urbana Weather Forecast and Conditions Illinois (61801)
Yahoo!
Make Y! your home page
Netflix: $4.99/mo. Movies delivered, no late fees
YAHOO!
Web Images Video Local Shopping more
Search:
Web Search
Yahoo! Home My Yahoo!
Feb 9, 2009 Page Options
Answers
Autos UPDATED!
Finance
Games
Groups
HotJobs
Maps
Mobile Web
Movies | TV
Music
OMG
Personals
Real Estate
Shine
Shopping
Featured Entertainment Sports Video
Stunning Super Bowl story
Ben Roethlisberger reveals that he played with a major injury when the Steelers won the title. » Details
• Fitzgerald wins MVP a week too late
• Ex-NFL star released from jail
QB's stunning revelation about the Super Bowl
Tiger Woods and wife welcome new baby
Companies that may not survive 2009
Best and worst dressed at the Grammy Awards
» More: Featured | Buzz
News World Local Finance
As of 1:05 p.m. CST
• Obama: Stimulus package is the right size and scope
• Australia bushfires: 'It is a fiery hailstorm from hell' | Photos
• SEC, Madoff agree to settle civil lawsuit for $50B Ponzi scheme
• Italy Senate pushing bill to force life support for coma victim
Check your mail status: Sign In
Free mail: Sign Up
Mail
Messenger
Puzzles
Weather
Events
Horoscopes
SUBWAY
$5 ANY
REGULAR FOOTLONG
ENTER SUB SHOWROOM
HURRY IN! LIMITED TIME ONLY
http://www.yahoo.com/s/1027905

The OP Browser

- Goal: build a secure web browser
- Provide an architecture for secure web browsing
  - Maintain security guarantees even when compromised
  - Integrate plugin policy into overall browser policy
- Use OS techniques, formal methods
  - Partition browser
  - Expose communication
  - Reason with formal methods
  - Analyze attacks

Gazelle

- Goal: improved display security
- More fine-grained isolation
  - Enables novel display security policies
- Trade off compatibility for security

OP2

- Culmination of recent work in secure browsers
  - Based on OP
  - Borrows ideas from Chrome and Gazelle
- Surprising preliminary performance results
  - Modifications for security improved performance

Outline

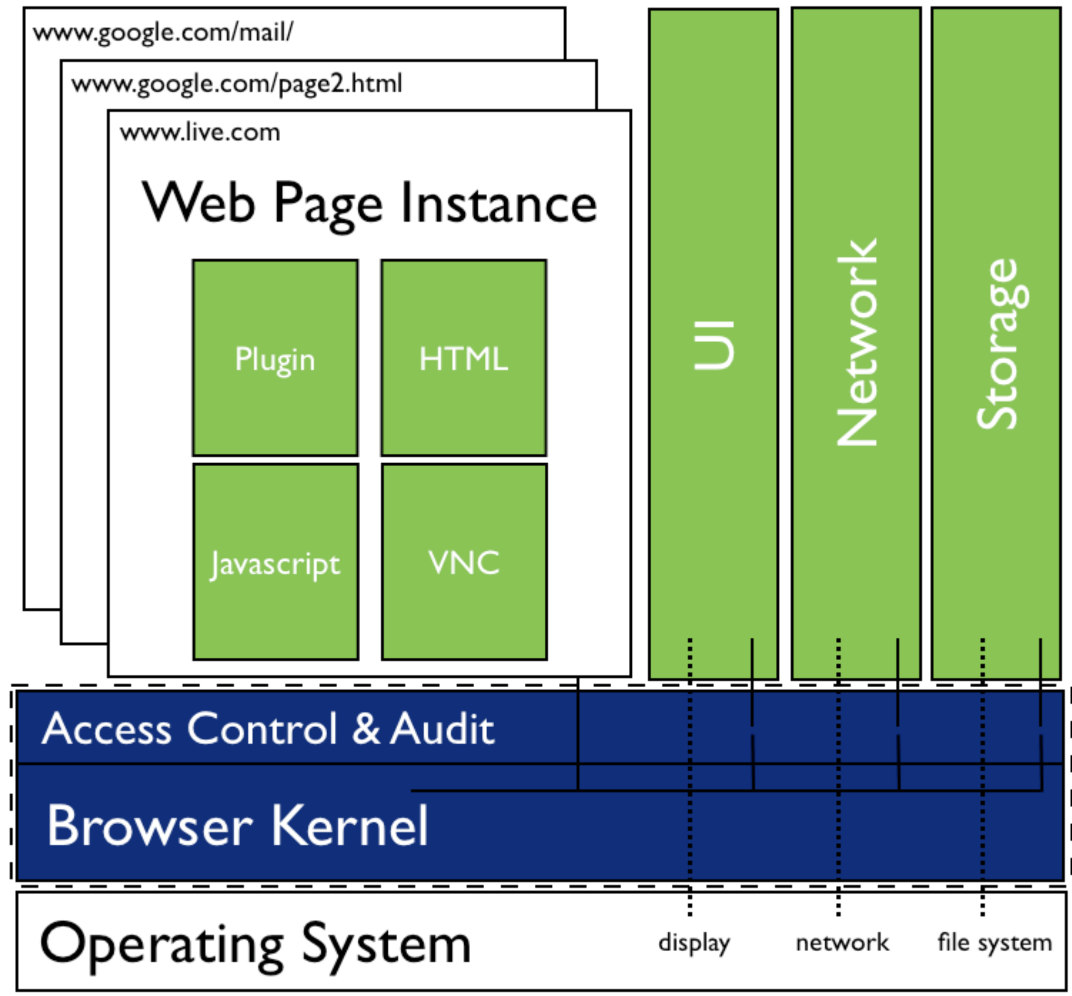
- OP browser design
  - Using formal methods to verify invariants
  - Performance
- OP2 and Gazelle
  - Display security
- **OP2 performance**
- Other research from my group

Threat Model

- Threat model: the attacker is targeting the browser and has complete control over content being served in the web page

OP design

- Decompose into browser subsystems
  - Web page instance further divided
- Use message passing
  - All messages through browser kernel
- Dedicated subsystems for OS operations
- Host OS sandboxing



Design enables security

- Partitioning and constrained communication enable new security mechanisms
  - Clean separation of browser functionality and security
- Policy
  - Easier to reason about current policies
  - Novel policies including for plugin security
- Formal methods
- Forensics

Use of formal methods

- Model using Maude
- Attack modeled by sending arbitrary messages

- Check SOP policy

- URL bar = URL loaded with compromise
  - Model checking revealed paths to bad state
  - Attacker could send out of order messages

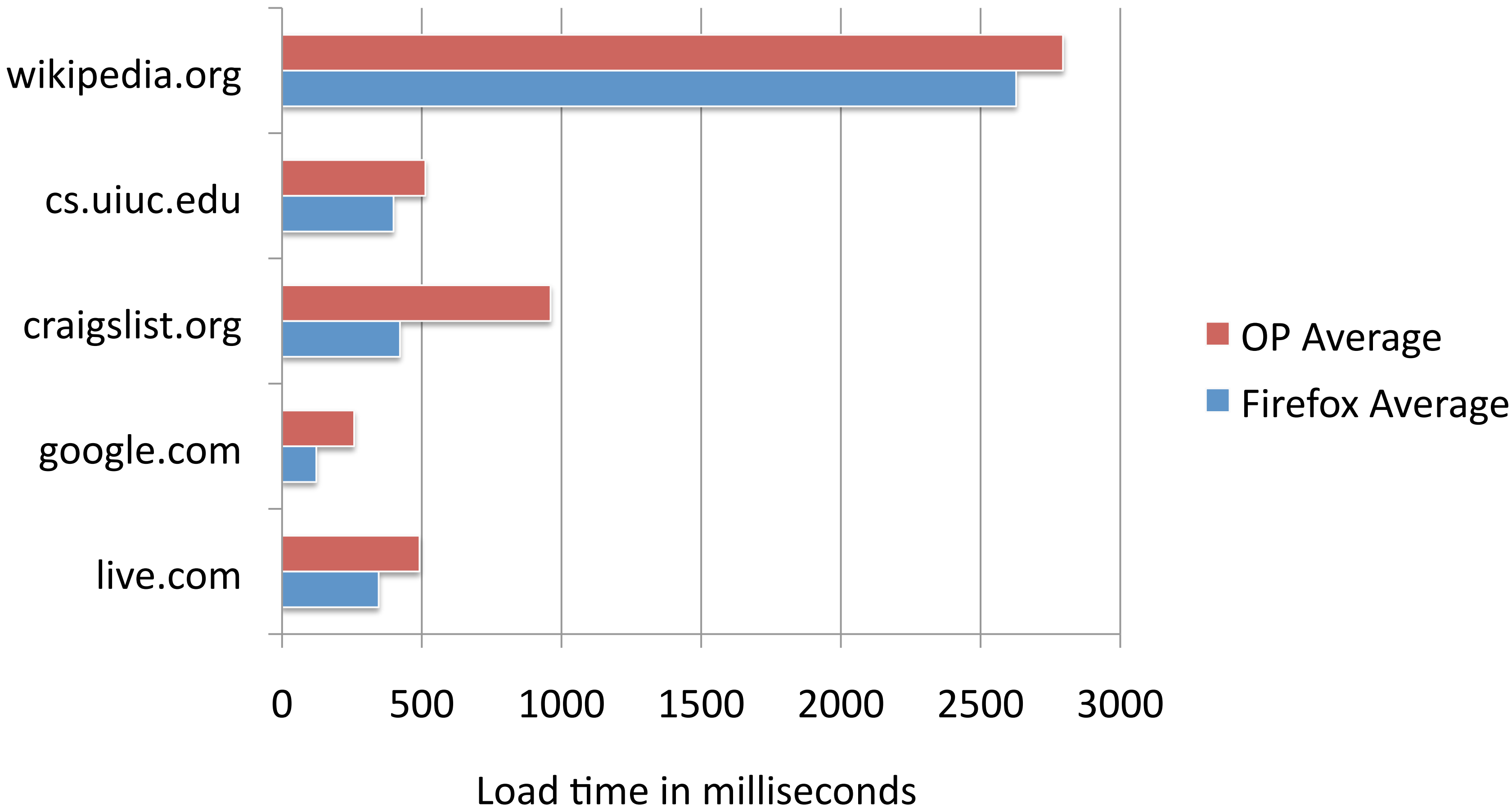
- Use to drive development
  - Fix bugs, update model, re-check

Implementation

- Use KHTML as rendering engine
- Rhino for JavaScript
- Use Java where it makes sense
  - C++ for browser kernel, only about 1000 LOC

Performance (circa 2007)

- Load latencies do not impact usability



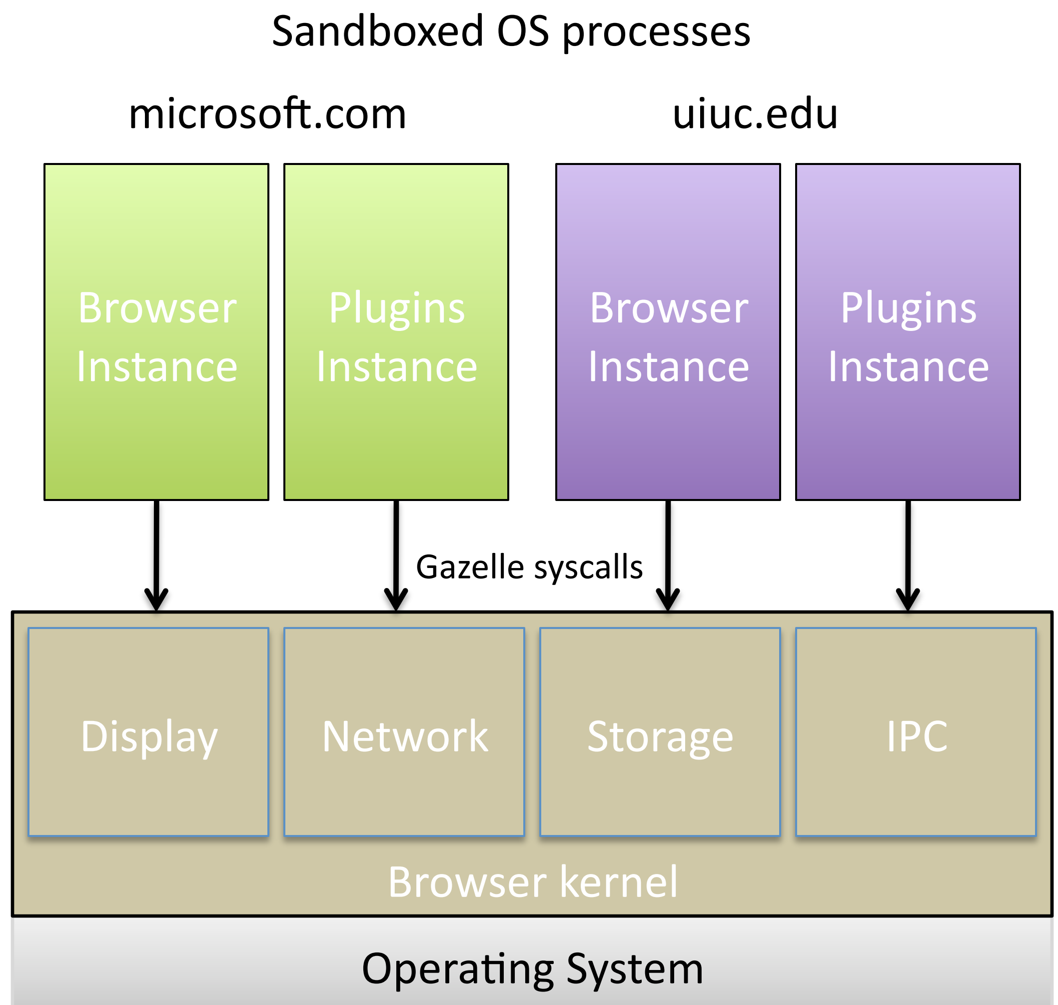
The Gazelle Web Browser

Helen J. Wang, Chris Grier, Alex Moshchuk,
Sam King, Piali Choudhury, and Herman Venter

Microsoft Research, UIUC, U. Washington

Gazelle architecture

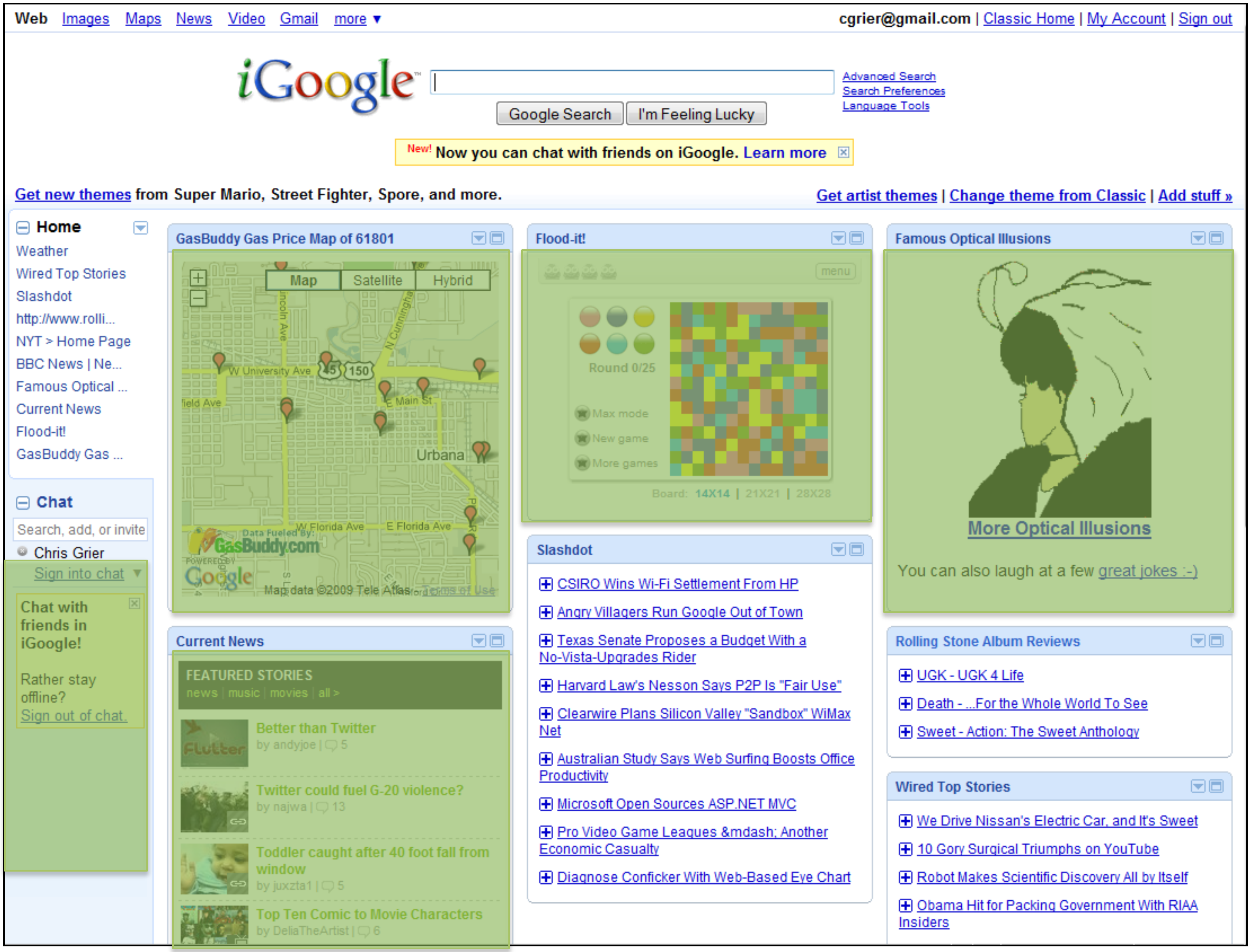
- Per-origin processes
  - Sandboxed
- Gazelle system calls for accessing resources
  - handled by browser kernel
- Browser instance is libc for web
  - Gazelle syscalls built-in
  - HTML handling
  - JS execution
- Browser Kernel
  - Access to system resources
  - Enforces all security policy



Display security in Gazelle

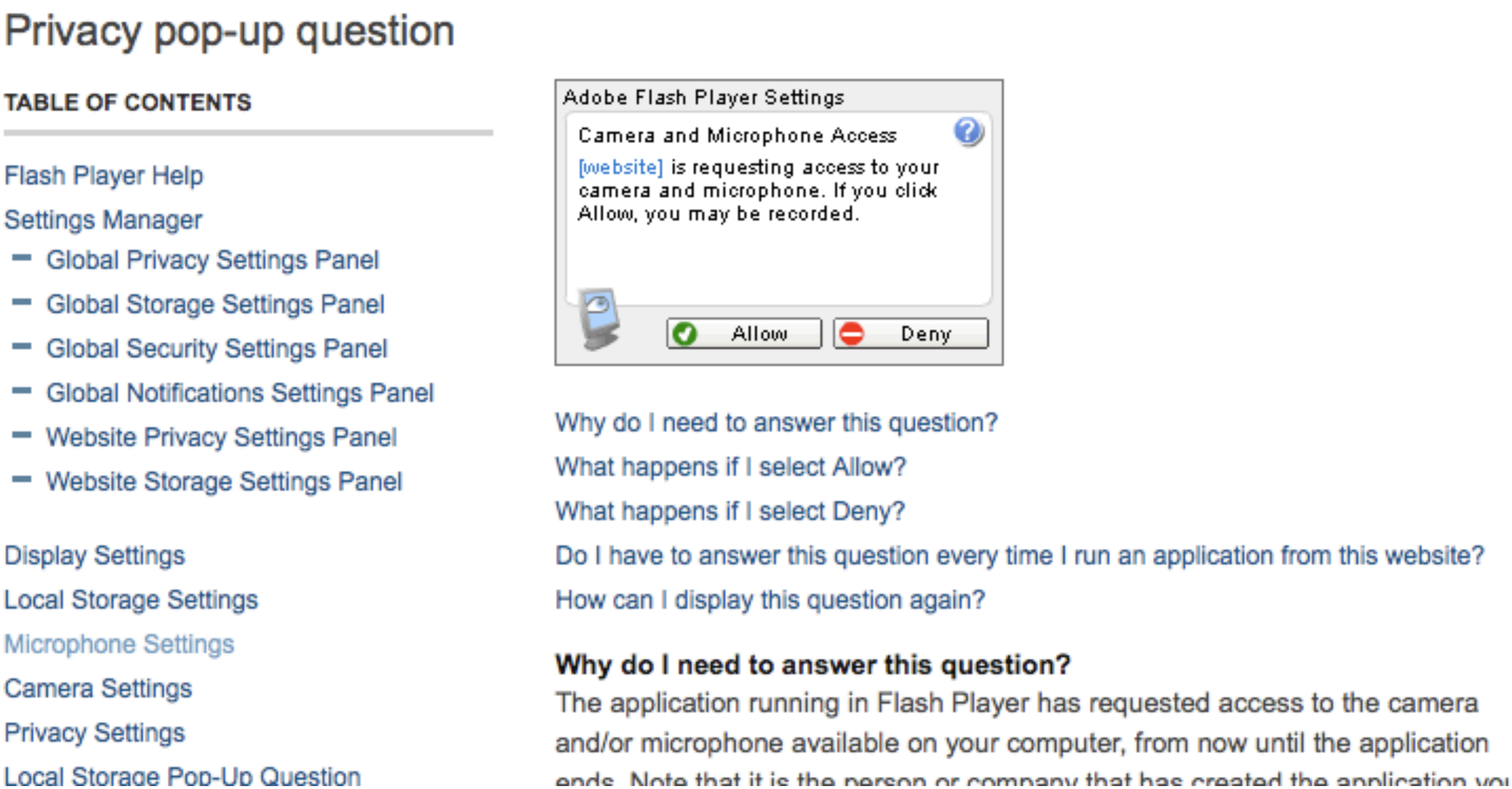
- Goal: Provide strong isolation between rendered content
- Compose content from many different services securely
- Not as clear decisions as traditional OS display
  - Cross principal content inherent in rendering on web
- Difficult cases can raise policy questions
  - Frames can be transparent
  - Images under text
  - Layers in CSS

What is display isolation?



- http://www.flickr.com/photos/24967759@N00/2924995732/

A website from adobe…



Privacy pop-up question

TABLE OF CONTENTS

Why do I need to answer this question?

What happens if I select Allow?

What happens if I select Deny?

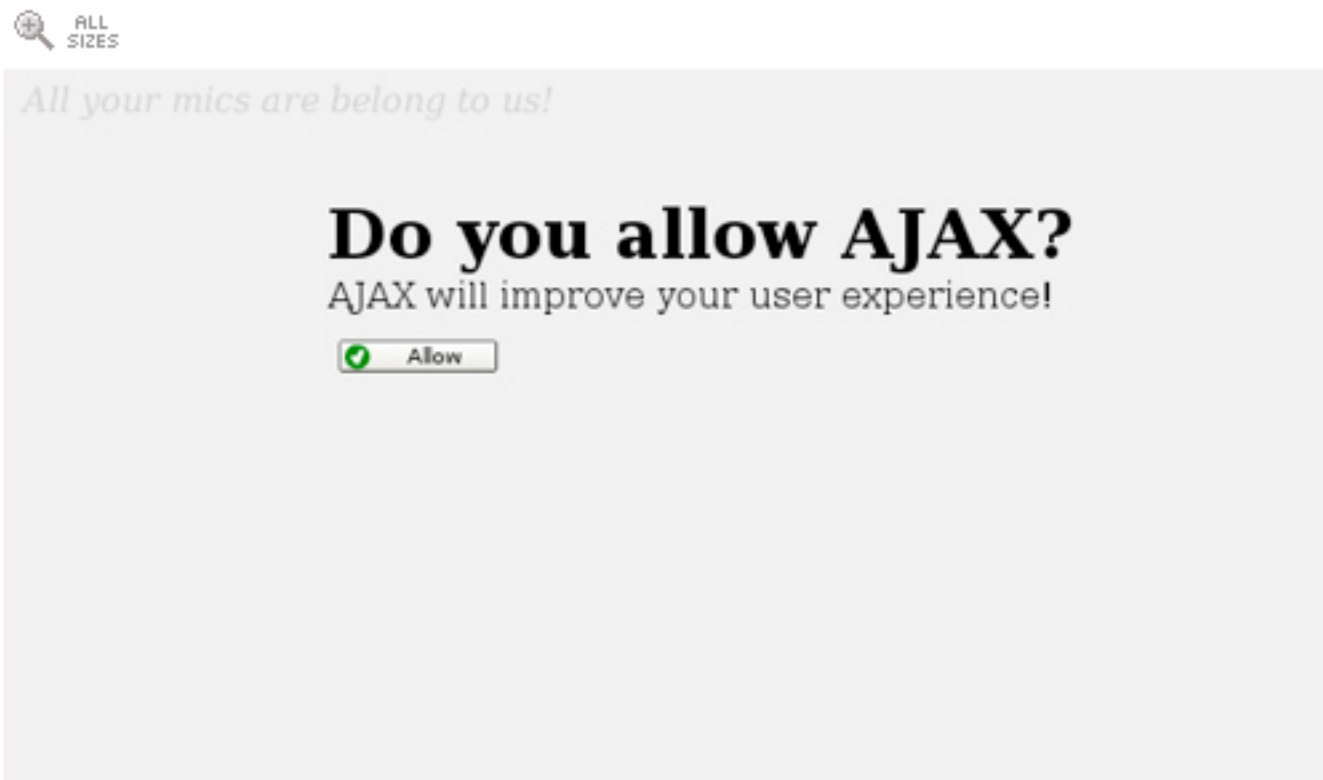
Do I have to answer this question every time I run an application from this website?

How can I display this question again?

Why do I need to answer this question?

The application running in Flash Player has requested access to the camera and/or microphone available on your computer, from now until the application ends. Note that it is the person or company that has created the application you

Redressing Flash



UI Redress attack against Flash

- http://www.flickr.com/photos/24967759@N00/2924995732/

Delegate once policy

- Delegate once policy
  - Delegate, screen space is lost for duration of the page
  - Cannot draw over or outside of delegated space
- Deviate from standards for improved security
- Prevents drawing over cross-origin content
- Helps (but doesn't eliminate) "UI redressing"
- Going to break certain things (menus, move ads, full scrn)

Display isolation mechanisms

- Enforce display policy in browser kernel
- Cross-domain iframes and plugins isolated
  - Rendered in separate processes

- **From paper, unclear if this will be practical**
  - Significant overhead for nytimes.com

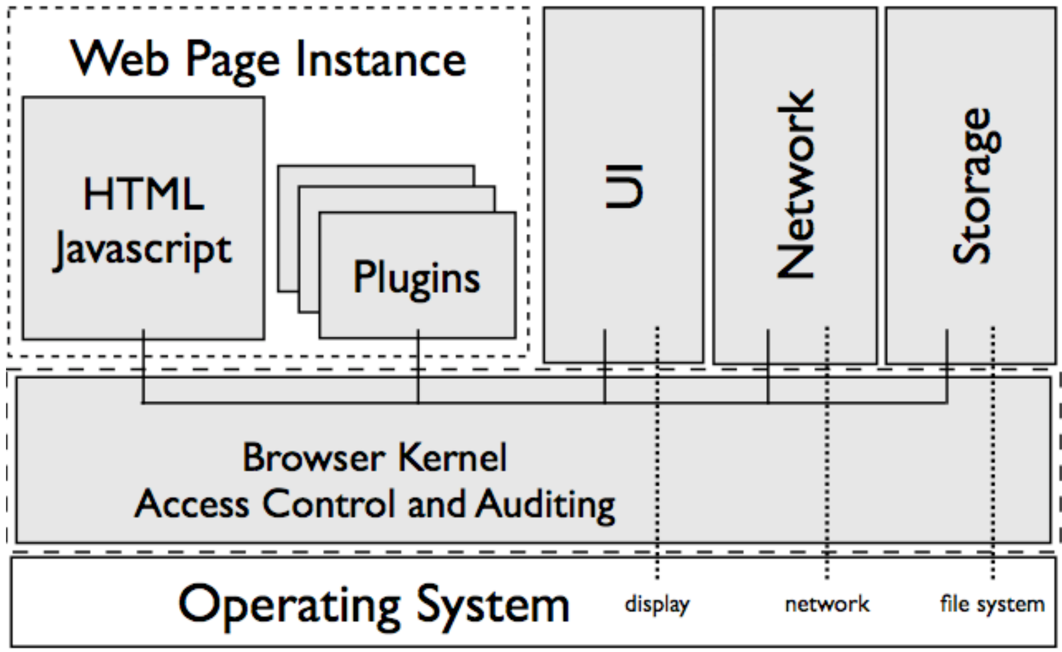
OP2: making secure browsers more practical

Shuo Tang, Chris Grier, **Sam King**

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

OP2



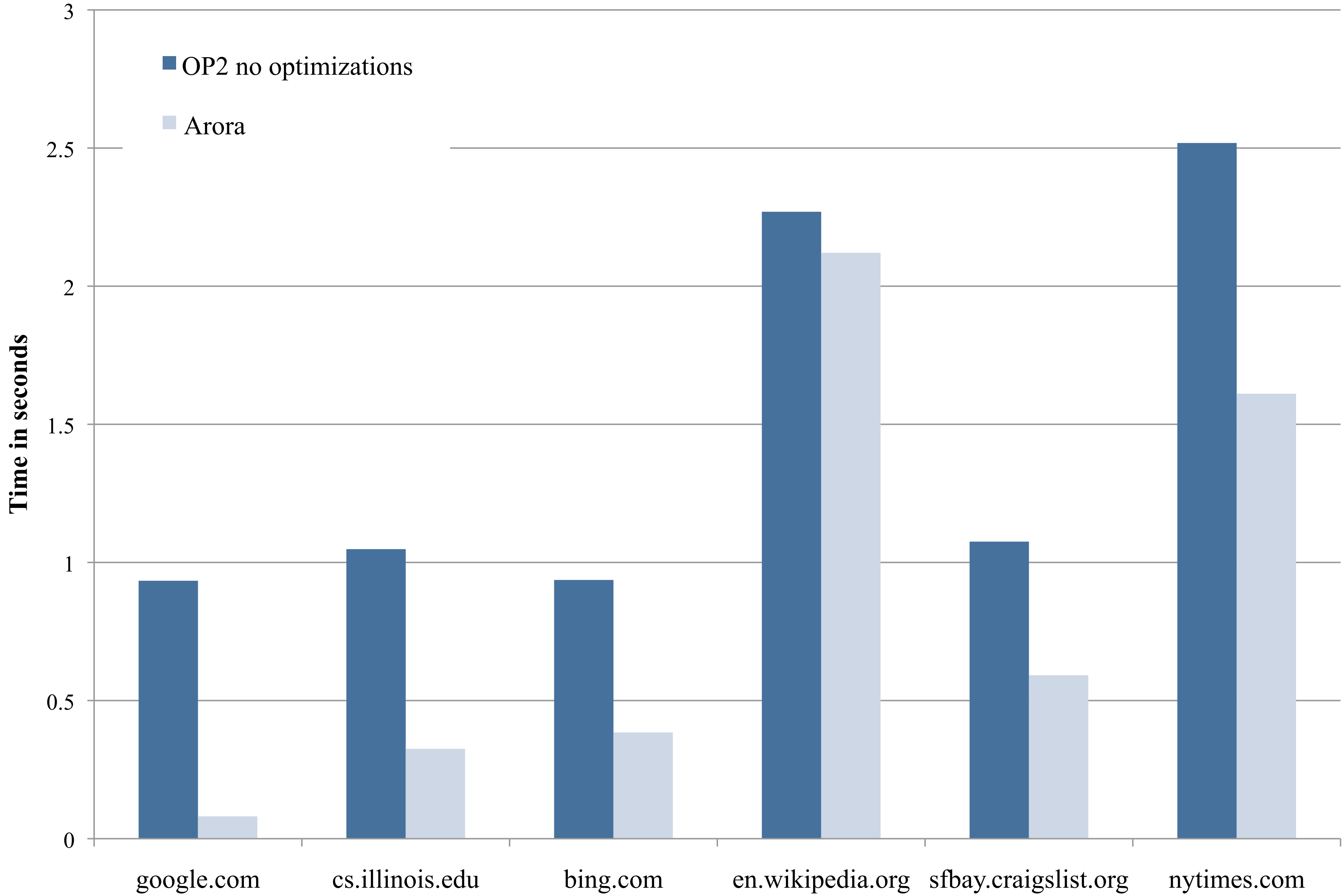
- Based on original OP browser
- From Chrome: combine JS and HTML rend.
  - Content sniffing algorithm from Barth *et al.*
- From Gazelle: display security mech and policy

OP2 implementation

- Implemented using WebKit and Qt
  - Linux and Mac version, results for Linux
  - Entire browser now in C++ to use Qt
  - Browser kernel about 1500 LOC
- Subtle diffs between OP2, Gazelle, and Chrome
  - OP2 reuse existing components when possible
  - Try to keep things as simple as possible
    - Fewer features
  - Less worried about compatibility
    - Try to maintain compat policies when possible

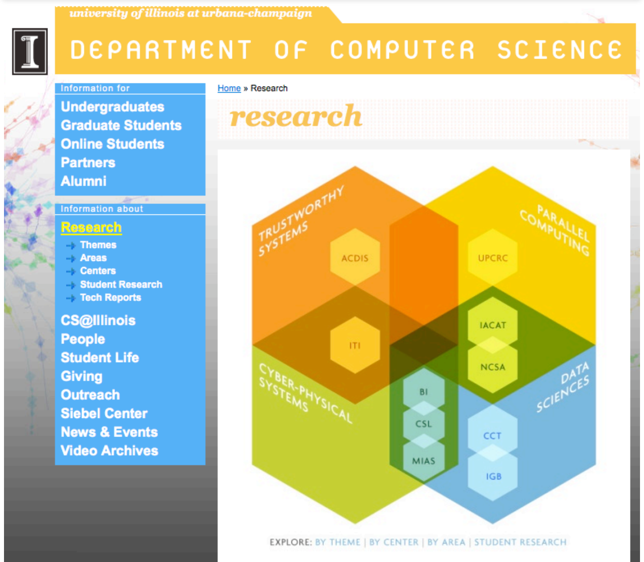
OP2 performance

- Performance experiments
  - Page load latency times, visit 10 times after warm
  - Compare vs latest version of Arora
    - Also uses WebKit and Qt, single process architecture
- Experimental setup
  - 2.66 GHz Core 2 Duo
  - 8 GB of RAM
  - Connected to school network



3
2.5
2
1.5
1
0.5
0
Time in seconds
OP2 no optimizations
Arora
google.com
cs.illinois.edu
bing.com
en.wikipedia.org
sfbay.craigslist.org
nytimes.com

http://berkeley.edu

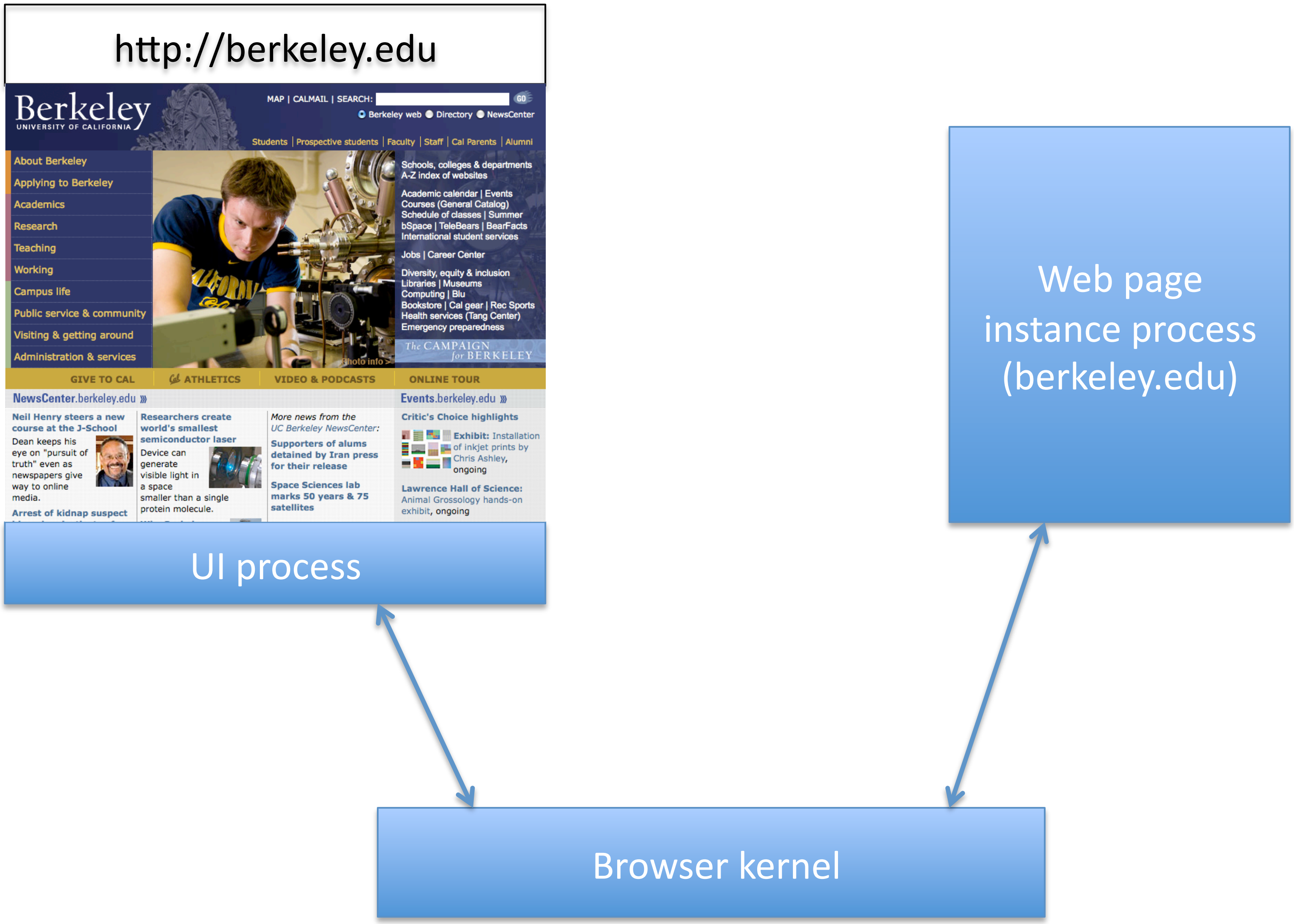


UI process

Web page instance process (cs.uiuc.edu)

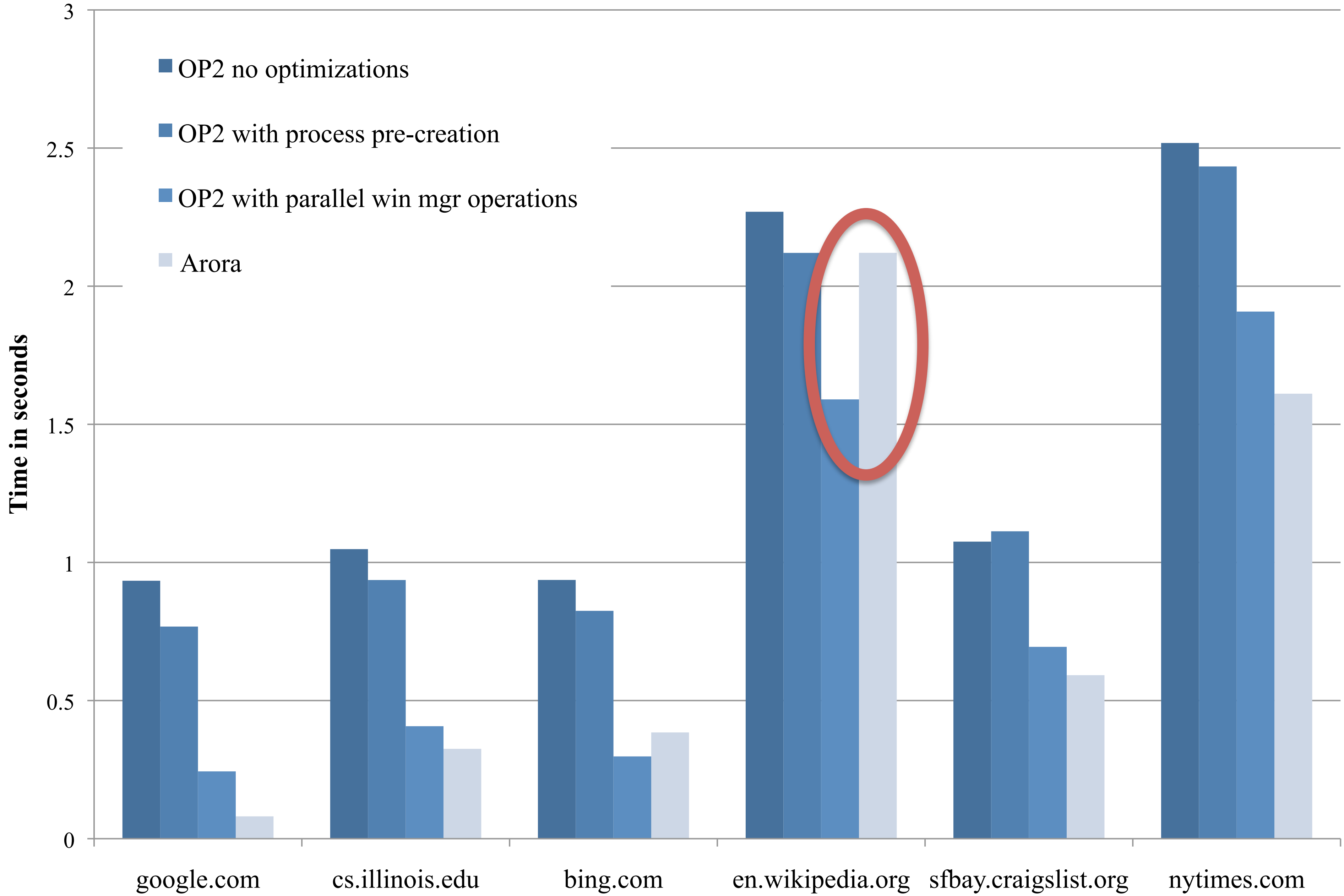
url = http://berkeley.edu

Browser kernel

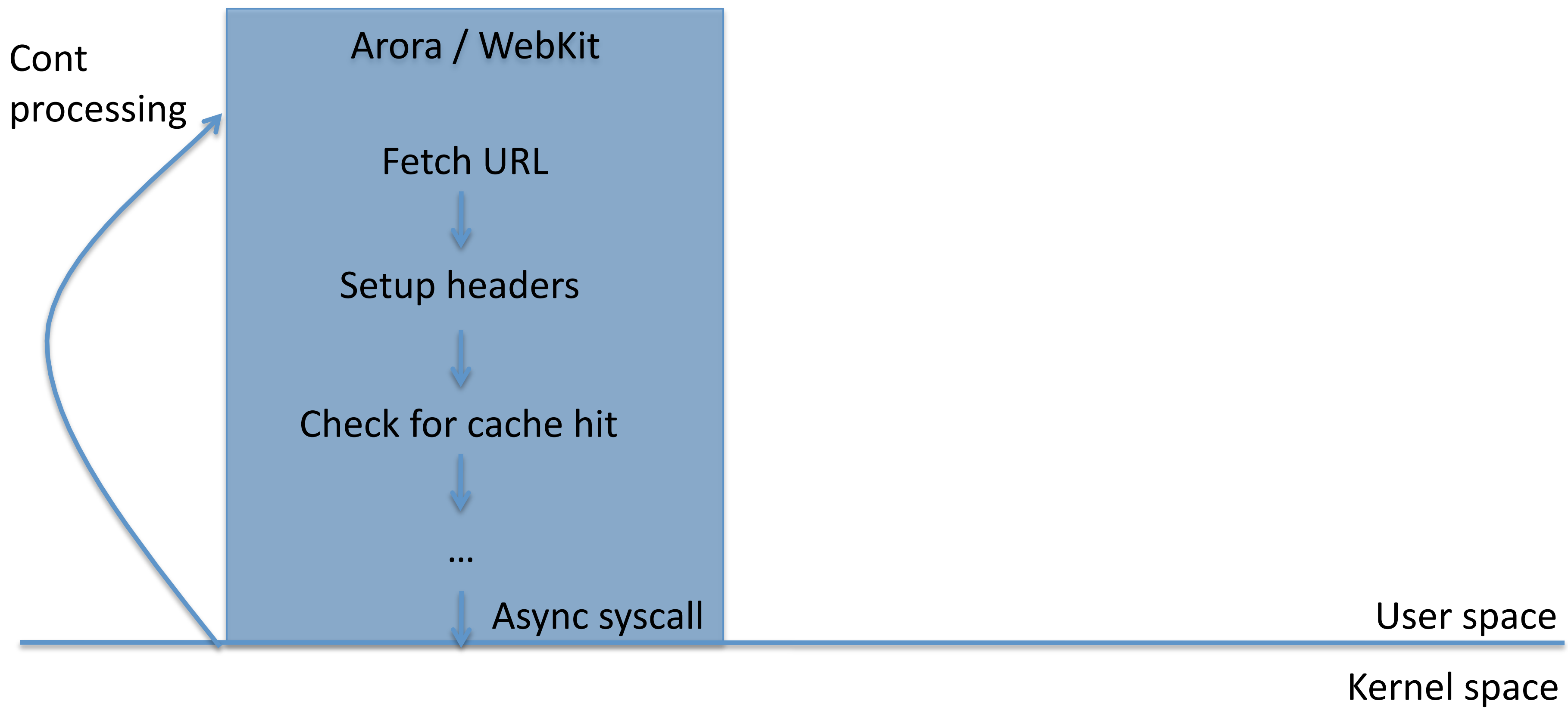

http://berkeley.edu
UI process
Web page instance process (berkeley.edu)
Browser kernel

Simple optimizations

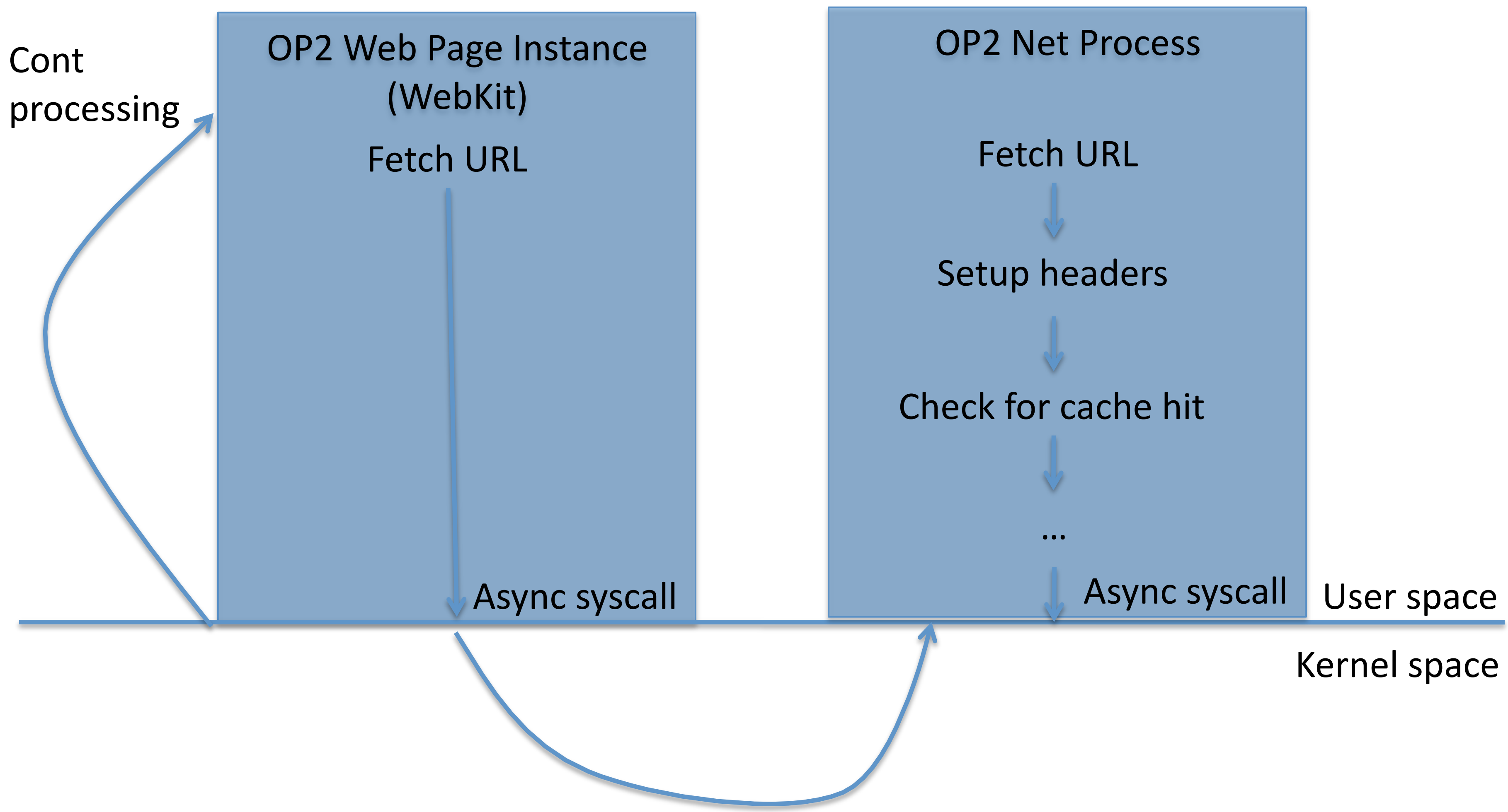
- Pre-create web page instance processes
- Overlap window mgr ops w page loading

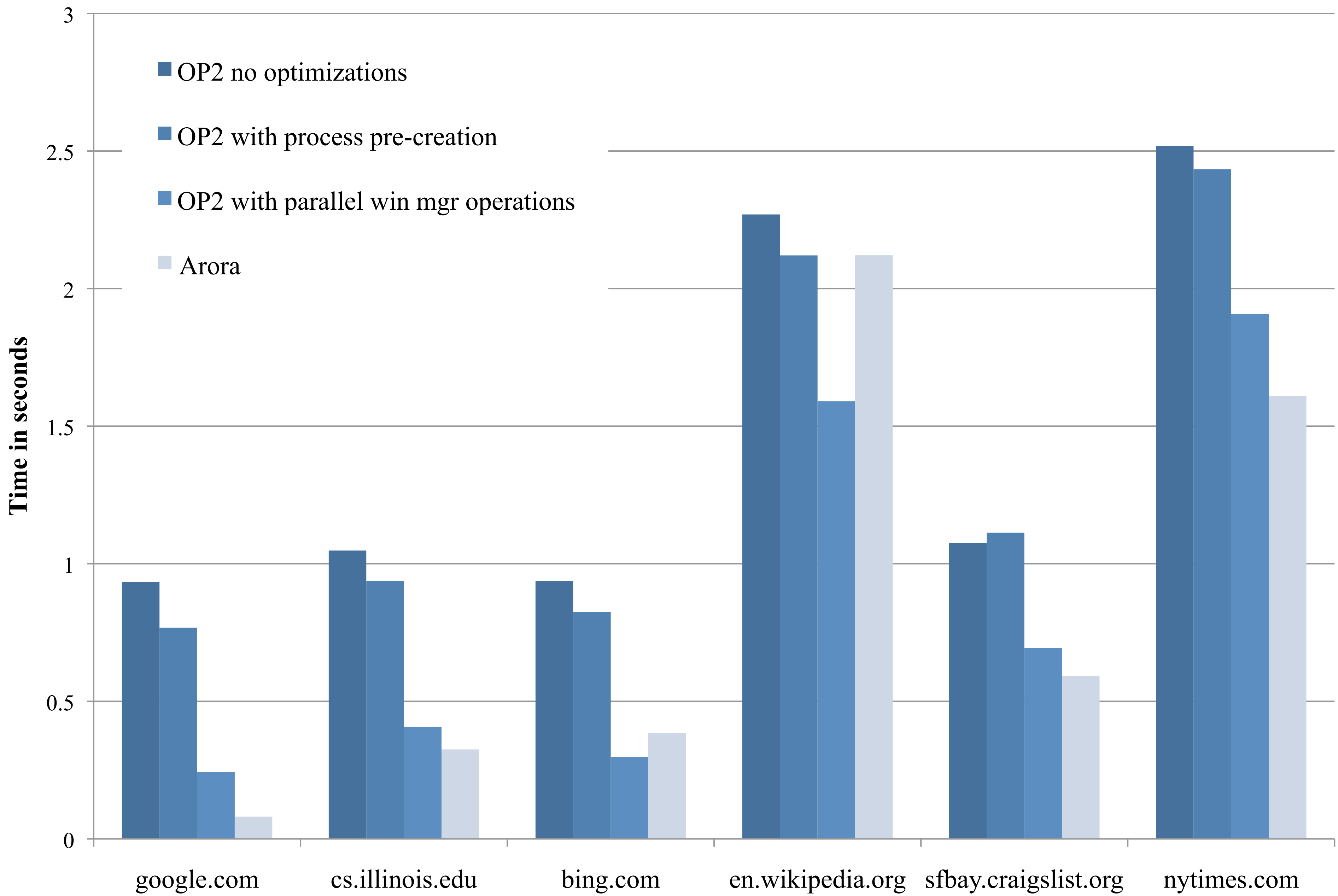

Time in seconds
0
0.5
1
1.5
2
2.5
3
OP2 no optimizations
OP2 with process pre-creation
OP2 with parallel win mgr operations
Arora
google.com
cs.illinois.edu
bing.com
en.wikipedia.org
sfbay.craigslist.org
nytimes.com

Adding parallelism in OP2



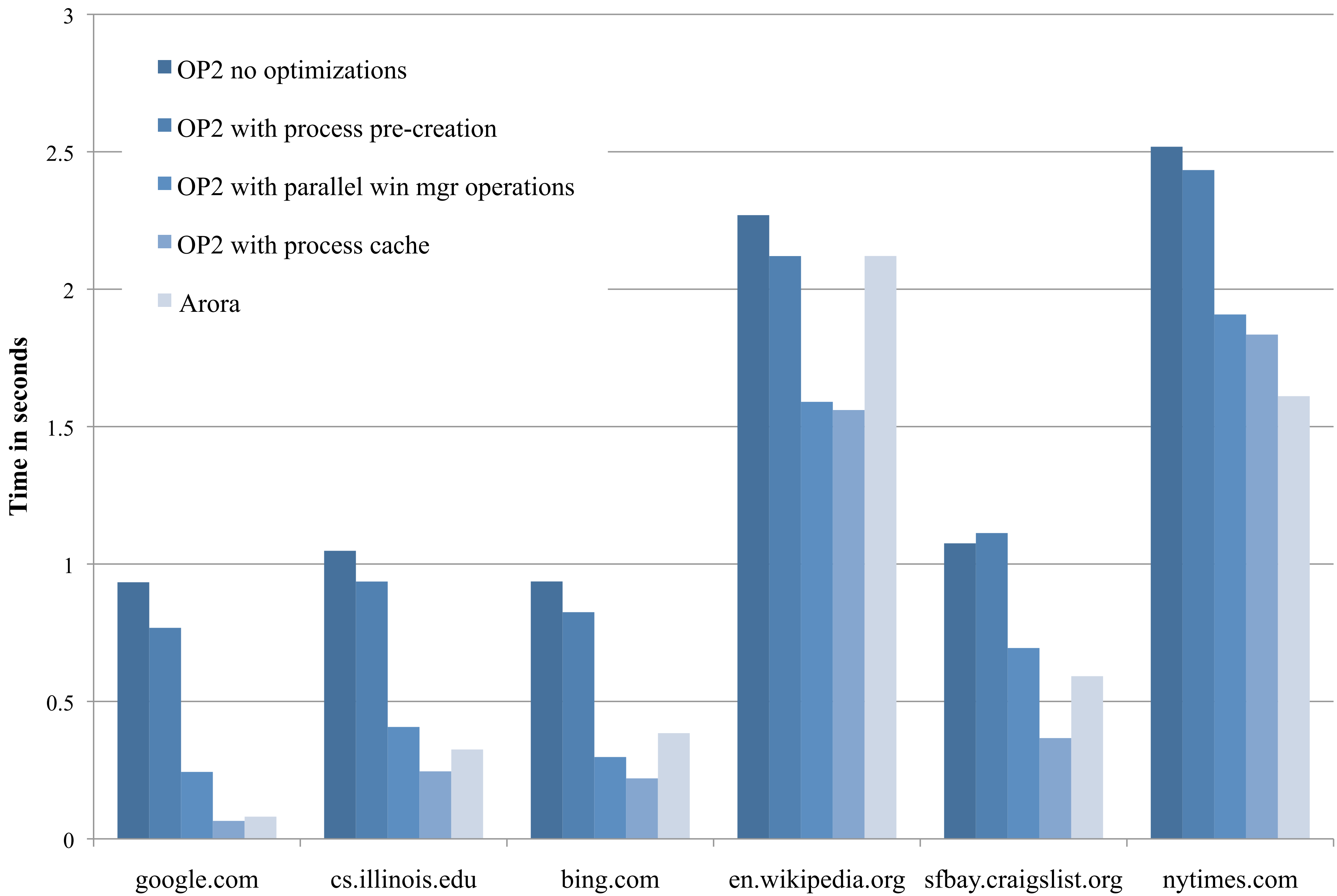
Adding parallelism in OP2




OP2 no optimizations
OP2 with process pre-creation
OP2 with parallel win mgr operations
Arora
Time in seconds
3
2.5
2
1.5
1
0.5
0
google.com
cs.illinois.edu
bing.com
en.wikipedia.org
sfbay.craigslist.org
nytimes.com

Process cache optimization

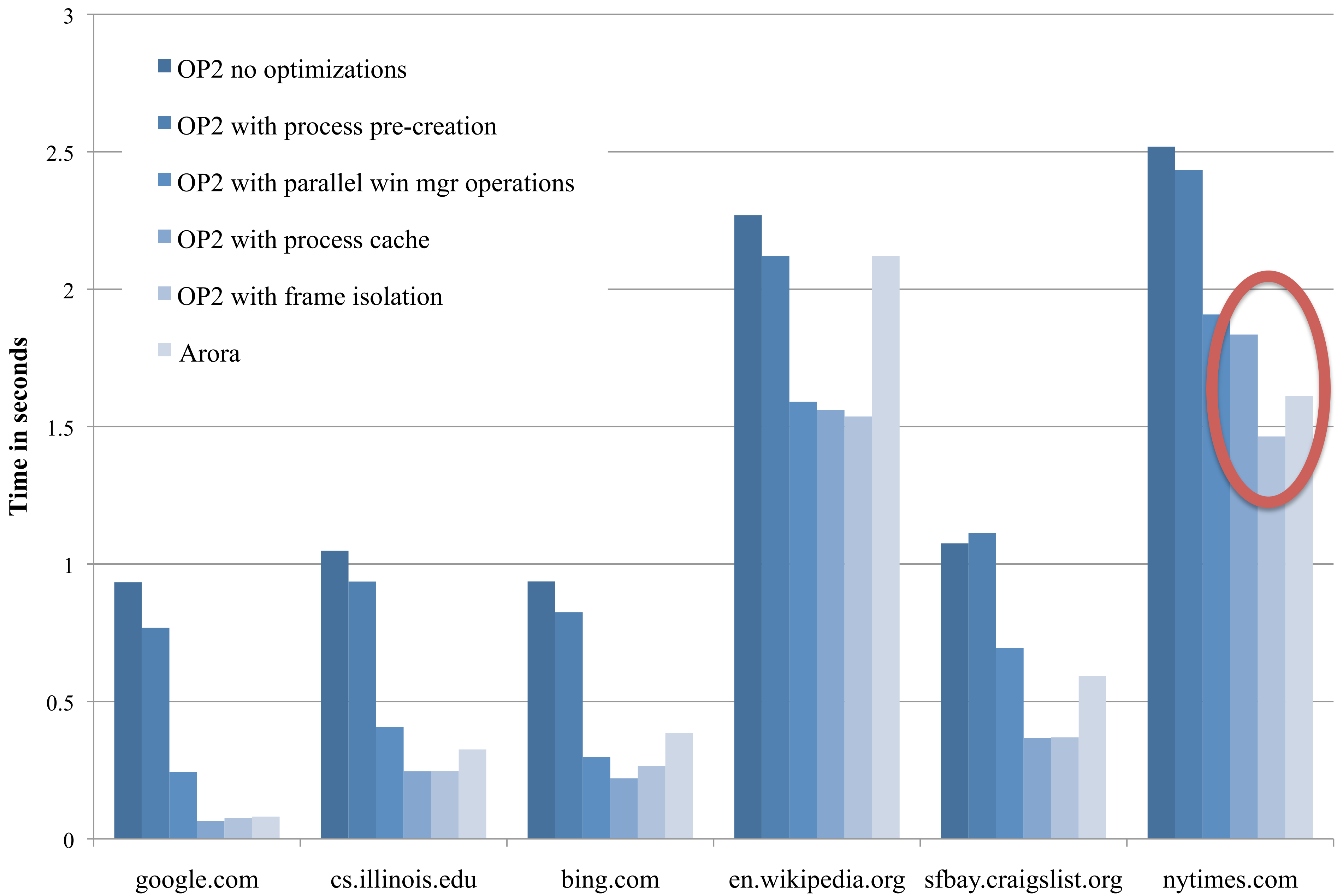
- WebKit built assuming process reuse
  - Cache web object in memory
- Starting from a fresh state fundamental to OP
  - Security purposes
- Solution: cache old web page instances
  - Hits only when we visit the exact same URL
  - Minimize amount of state that could be leaked



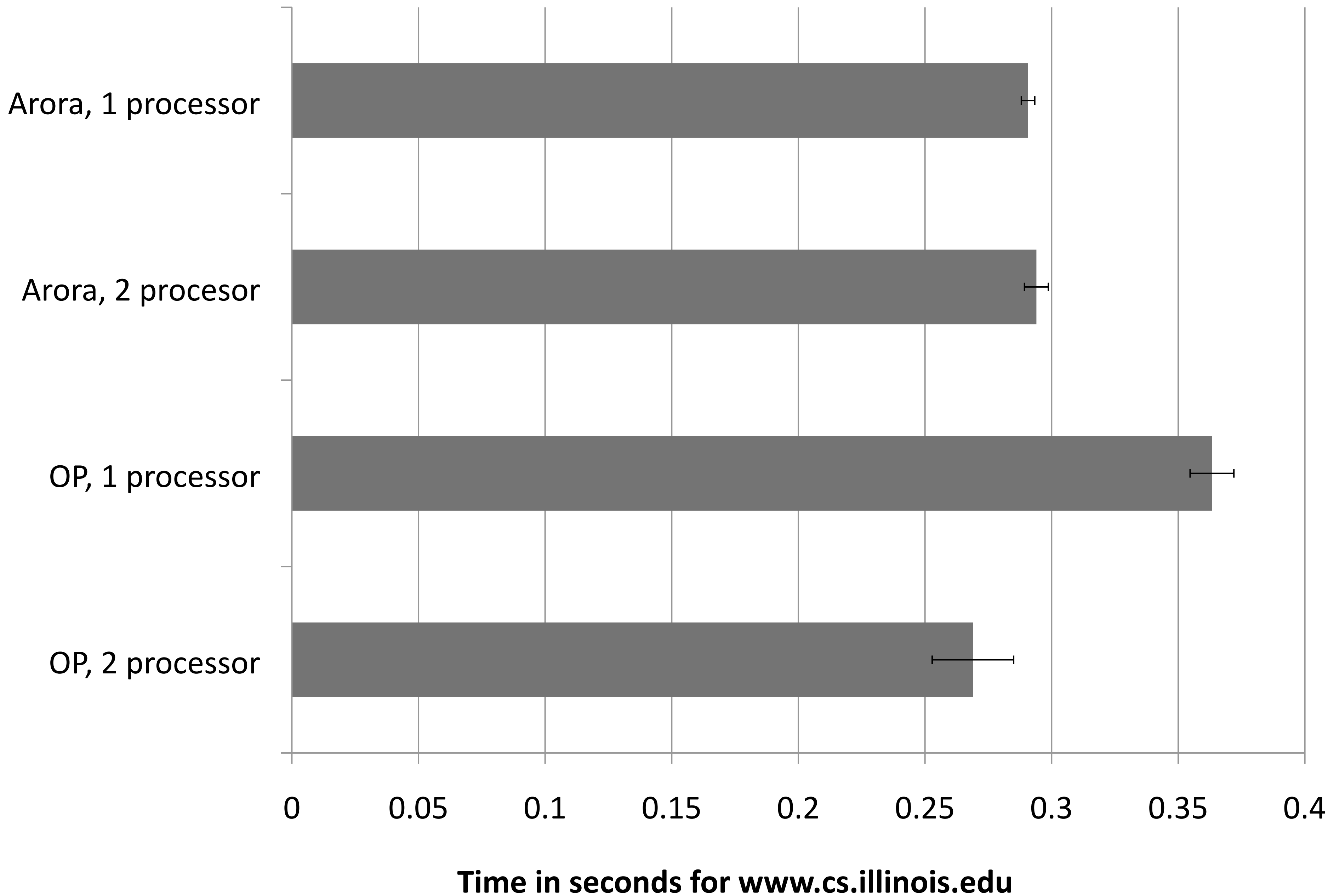
OP2 no optimizations
OP2 with process pre-creation
OP2 with parallel win mgr operations
OP2 with process cache
Arora
Time in seconds
3
2.5
2
1.5
1
0.5
0
google.com
cs.illinois.edu
bing.com
en.wikipedia.org
sfbay.craigslist.org
nytimes.com

Display isolation mechanisms

- Have a fully optimized OP2 browser
  - Determine if display isolation could be practical
- Put cross origin iframes in separate processes
  - Done for security reasons, easier to label
  - Can enforce display policies in OP2 browser kern.



OP2 no optimizations
OP2 with process pre-creation
OP2 with parallel win mgr operations
OP2 with process cache
OP2 with frame isolation
Arora
Time in seconds
0
0.5
1
1.5
2
2.5
3
google.com
cs.illinois.edu
bing.com
en.wikipedia.org
sfbay.craigslist.org
nytimes.com


Arora, 1 processor
Arora, 2 procesor
OP, 1 processor
OP, 2 processor
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Time in seconds for www.cs.illinois.edu

Lessons learned from OP2 eval

- Changes for security improved performance
  - Usually shoot for 100% overhead or less
    - 25% don't even have to explain
- Huge opportunities for performance gains
  - Performance optimizations for architecture
  - Accidentally improved performance

Related Work

- New architectures
  - Using VMMs: Tahoma [Oakland ‘06]
  - File system focused: *Building a secure web browser* [FREENIX ‘01]
  - Process based: *Architectural principles for safe web programs* [HOTNETS ’07]
- Securing existing applications and new abstractions
  - Javascript: Browsershield, JS Instrumentation [OSDI ‘06, POPL ’07]
  - Privacy: Safecache/Safehistory [WWW ‘06]
  - SOP: locked SOP, Script accenting [CCS ‘07]
- Formal methods
  - UI invariants: *A systematic approach to uncover security flaws in gui logic [Oakland ’07]*
- Other attacks

Conclusions

- Treat browser like an OS, more secure
- OP and OP2 improve security
  - OP2 also improves performance

- A step towards preventing, containing, and recovering from browser-based attacks
- A step towards a parallel web browser

Questions?

- (Note: this is not the end of my talk yet)

Untrusted computing base: defending against malicious hardware

Matt Hicks, Matt Finnicum, **Sam King**
University of Illinois

Milo M.K. Martin and Jonathan Smith
University of Pennsylvania

Building secure systems

- We make assumptions when designing secure systems
- Break secure system, break assumptions
  - E.g., look for crypto keys in memory
- People assume hardware is correct

- What if we break this assumption?

Malicious hardware

- Is it possible to modify design of processors?

- Implementing hardware is difficult
- Implementing HW-based attacks is easy!
  - Small hardware level *footholds*
  - Execute high-level high-value attacks WITHOUT exploiting any software bugs

Defenses

- Based on insights from foothold devel.
- Analyze circuit at design time
- Highlight potentially malicious circuits

- Hope to have results soon

Deterministic replay

- Record execution, reproduce arbitrary past states
  - Debugging, fault tolerance, security, etc.
- SW-only replay flexible, slow for MP machines
- HW-only fast, record entire machine
  - Less flexible for current uses
- Combine HW and SW replay
  - Naïve approach does not work
  - Subtle and fundamental issues
- Capo: HW/SW interface and abstractions for record and replay
- Paper in ASPLOS '09
  - Joint work with Pablo Montesinos, Matt Hicks, and Josep Torrellas

Digging for Data Structures

Anthony Cozzie, Frank Stratton, Hui Xue, Sam King
University of Illinois at Urbana-Champaign

Data Structure based Antivirus

- Detect programs based on their data structs
- Convert seemingly random bytes of program memory into data structures automatically
  - Mark each word as an int, pointer, string, etc.
  - Use Bayesian classifier – see paper for details
- Two programs with same data structures are likely the same program
  - Worked for three modern botnets
- Presented at OSDI '08

Other projects

- Automatic fault recovery
  - Paper in ASPLOS '09
  - Joint work with Andrew Lenharth and Vikram Adve
- More secure web browser work

Questions?

- (Note: this **is** the end of the talk)

Replicate portions of the OS

- Extracts parts of OS needed for web client sec
  - Custom labeling and access control system
  - RPC / message passing layer
  - Window manager (limited extent)

Assumptions about OS

- Process-level isolation
  - Memory protection
  - well-known IPC mechanisms
- System-level sandboxing
  - Isolate processes from system resources
  - Restrict system call capabilities
- Resource management
  - Create processes, message forwarding and naming
  - Network, disk, screen

Differences between OP, Chrome, Gazelle and OP2

Browser	Kernel	Nav.	Sub-windows	Frames	Display policy	Display mech.
OP	microkernel	isolated	isolated	not isolated	none	streaming image
Chrome	monolithic	isolated	different-site isolated	not isolated	none	custom
Gazelle	monolithic	different-origin isolated	isolated	different-origin isolated	opaque overlay	streaming image
OP2	microkernel	isolated	isolate	different-origin isolated	delegate once	window manager