QA: Dynamic Analysis & Advanced Testing

17-313 Fall 2024

Foundations of Software Engineering

https://cmu-313.github.io

Michael Hilton and Rohan Padhye

Administrivia

- P3 due Thursday, Oct 31st
  - Still facing deployment issues? See Slack or come to OH
  - Tools not passing CI with green checkmark?
    - First try to find ways to fix or suppress warnings and document those in the design doc
    - Otherwise, submit link to workflow where tool runs successfully even if there is a red cross for unfixed warnings and justify in the design doc (see: “integration” section)
- Next Thursday: Guest lecture

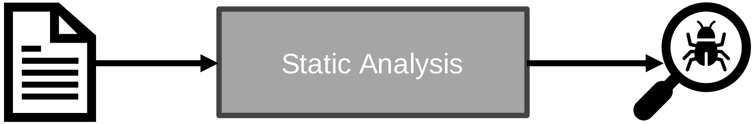
Smoking Section

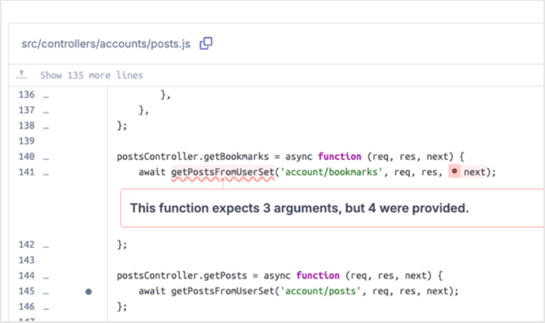
- Last **two** full rows

Learning Goals

- Describe random test-input generation strategies such as fuzz testing
- Identify and discuss the key challenges associated with performance testing in software development.
- Understand the ideas behind chaos engineering and how it is used to test resiliency of cloud-based applications
- Describe A/B testing for usability
- Recommend appropriate dynamic analysis techniques for specific software quality issues.

Recap: Program Analysis Tools

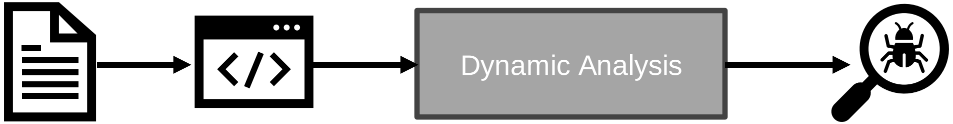


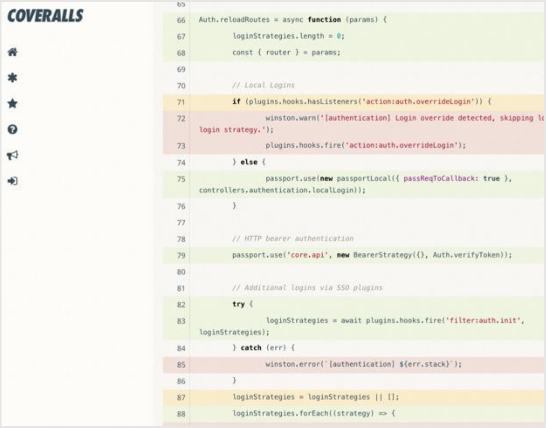

src/controllers/accounts/posts.js

```
Show 135 more lines
            },
        },
    };

    postsController.getBookmarks = async function (req, res, next) {
        await getPostsFromUserSet('account/bookmarks', req, res, next);
```

This function expects 3 arguments, but 4 were provided.

```
    };

    postsController.getPosts = async function (req, res, next) {
        await getPostsFromUserSet('account/posts', req, res, next);
    };
```





```
Auth.reloadRoutes = async function (params) {
        loginStrategies.length = 0;
        const { router } = params;

        // Local Logins
        if (plugins.hooks.hasListeners('action:auth.overrideLogin')) {
                winston.warn('[authentication] Login override detected, skipping lo
login strategy.');
                plugins.hooks.fire('action:auth.overrideLogin');
        } else {
                passport.use(new passportLocal({ passReqToCallback: true },
controllers.authentication.localLogin));
        }

        // HTTP bearer authentication
        passport.use('core.api', new BearerStrategy({}, Auth.verifyToken));

        // Additional logins via SSO plugins
        try {
                loginStrategies = await plugins.hooks.fire('filter:auth.init',
loginStrategies);
        } catch (err) {
                winston.error(`[authentication] ${err.stack}`);
        }
        loginStrategies = loginStrategies || [];
        loginStrategies.forEach((strategy) => {
```

Automated Analysis for Functional and Non-Functional Properties

- Correctness – Static Analysis and Testing
- Robustness – Fuzzing
- Performance – Profiling
- Scalability – Stress testing
- Resilience – Soak testing
- Reliability – Chaos Engineering
- Usability – A/B testing

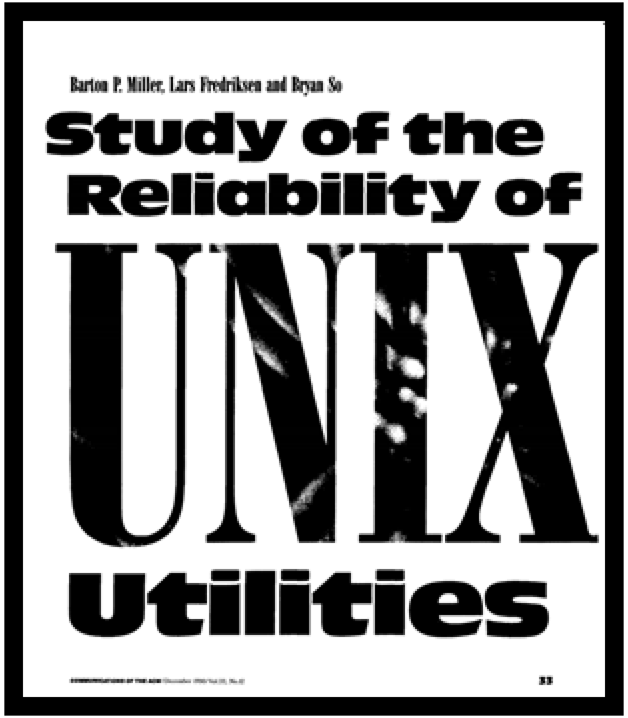
Automated Analysis for Functional and Non-Functional Properties

- Correctness – Static Analysis and Testing
- **Robustness – Fuzzing**
- **Performance – Profiling**
- **Scalability – Stress testing**
- **Resilience – Soak testing**
- **Reliability – Chaos Engineering**
- **Usability – A/B testing**

Outline

- Fuzz Testing
- Performance Testing and Debugging
- Testing in Production
  - Reliability: Chaos Engineering
  - GUI and Usability: A/B Testing

Security and Robustness



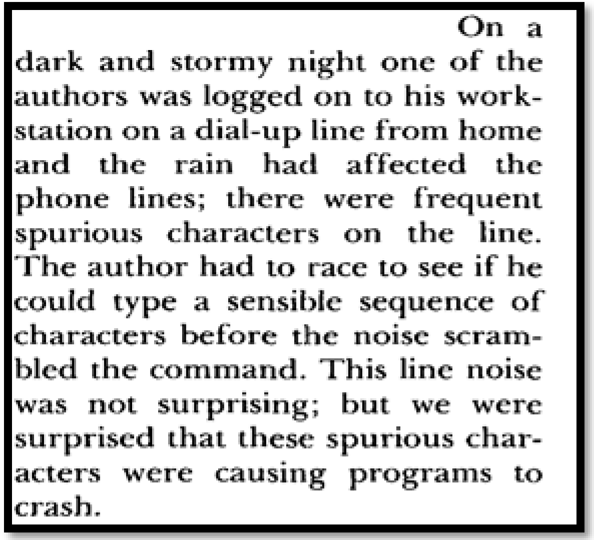
Barton P. Miller, Lars Fredriksen and Bryan So

Study of the Reliability of UNIX Utilities

[illegible] 33

Communications of the ACM (1990)

“



On a dark and stormy night one of the authors was logged on to his workstation on a dial-up line from home and the rain had affected the phone lines; there were frequent spurious characters on the line. The author had to race to see if he could type a sensible sequence of characters before the noise scrambled the command. This line noise was not surprising; but we were surprised that these spurious characters were causing programs to crash.

”

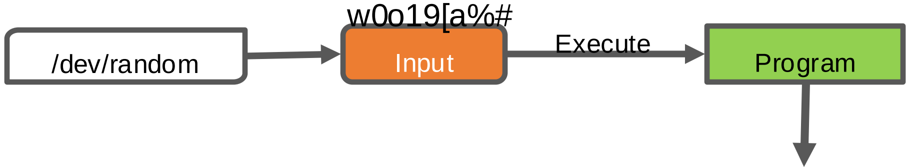
How to identify these bugs?

Infinite monkey theorem

*“a **monkey** hitting keys **at random** on a typewriter **keyboard** for an **infinite amount of time** will almost surely type any given text, including the complete works of **William Shakespeare.** “*

https://en.wikipedia.org/wiki/Infinite_monkey_theorem

Fuzz Testing



🐞

A 1990 study found crashes in:
adb, as, bc, cb, col, diction, emacs, eqn, ftp, indent, lex, look, m4, make, nroff, plot, prolog, ptx, refer!, spell, style, tsort, uniq, vgrind, vi

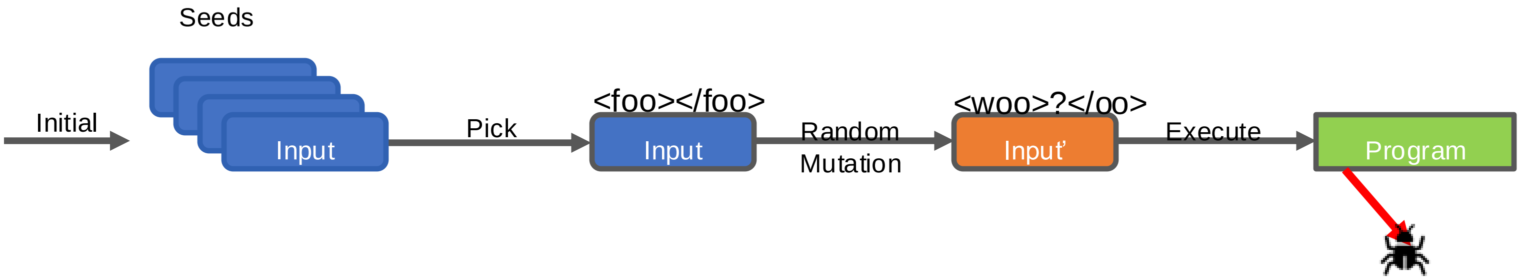
Common Fuzzer-Found Bugs in C/C++

Causes: incorrect arg validation, incorrect type casting, executing untrusted code, etc.

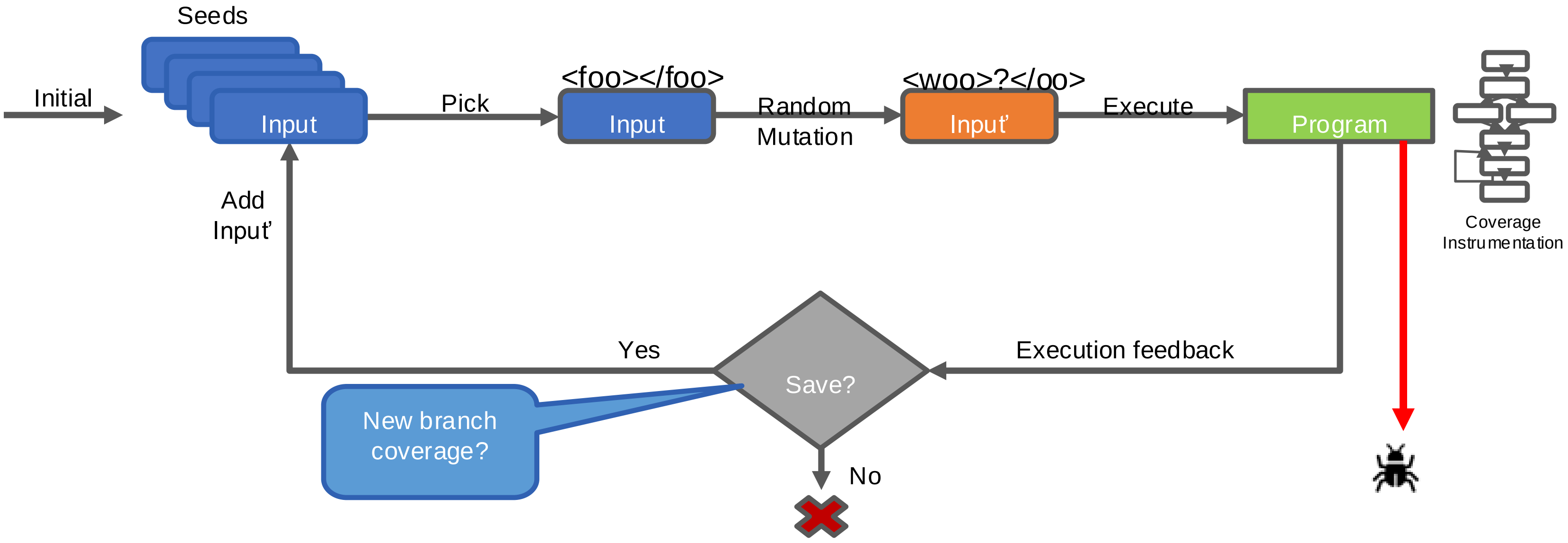
Effects: buffer-overflows, memory leak, division-by-zero, use-after-free, assertion violation, etc. (“crash”)

Impact: security, reliability, performance, correctness

Mutation-Based Fuzzing (e.g. Radamsa)



Coverage-Guided Fuzzing (e.g. AFL)



Mutation Heuristics

- Binary input
  - Bit flips, byte flips
  - Change random bytes
  - Insert random byte chunks
  - Delete random byte chunks
  - Set randomly chosen byte chunks to *interesting* values e.g. INT_MAX, INT_MIN, 0, 1, -1, …
- Text input
  - Insert random symbols relevant to format (e.g. “<“ and “>” for xml)
  - Insert keywords from a dictionary (e.g. “<project>” for Maven POM.xml)
- GUI input
  - Change targets of clicks
  - Change type of clicks
  - Select different buttons
  - Change text to be entered in forms
  - … Much harder to design

Fuzzing in practice

- Google uses ClusterFuzz to fuzz all Google products
- Supports multiple fuzzing strategies
- *“As of February 2023, ClusterFuzz has found ~27,000 bugs in Google (e.g. Chrome).”*
    - More than 50% of all bugs in the database
- Many bugs can also be fixed automatically
    - Continuous fuzzing and fixing
- New: Fuzz-driven development

Activity:

Pick one scenario based on where you are seating

- E-Commerce Web Application (front rows)
- Automotive Software for Self-Driving Cars (middle rows)
- Mobile Gaming Application (back rows)

Discuss in groups of 2-3 the applicability of fuzz testing in your scenario, considering:

- Types of inputs to fuzz.
- Potential vulnerabilities or bugs fuzz testing might uncover.
- Specific challenges in implementing fuzz testing for the scenario.

Bonus: How fuzz testing could be integrated into the development cycle for that particular application?

Performance Testing and Debugging

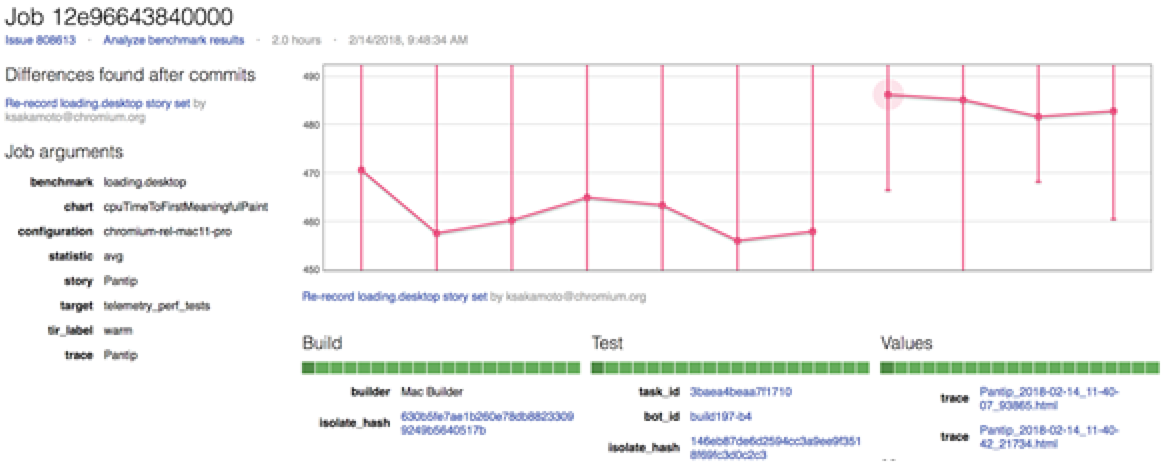
Performance testing: challenging requirements

- Goal: Identify *performance bugs*. What are these?
  - Unexpected bad performance on some subset of inputs
  - Performance degradation over time
  - Difference in performance across versions or platforms

- Not as easy as functional testing. What's the baseline?
  - Fast = good, slow = bad // but what's the threshold?
  - How to get reliable measurements?
  - How to debug where the issue lies?

Performance regression testing identifies trends

- Measure execution time of critical components
- Log execution times and compare over time



Job 12e96643840000

Issue 808613 · Analyze benchmark results · 2.0 hours · 2/14/2018, 9:48:34 AM

Differences found after commits

Re-record loading.desktop story set by ksakamoto@chromium.org

Job arguments

benchmark	loading.desktop
chart	cpuTimeToFirstMeaningfulPaint
configuration	chromium-rel-mac11-pro
statistic	avg
story	Pantip
target	telemetry_perf_tests
tir_label	warm
trace	Pantip

Re-record loading.desktop story set by ksakamoto@chromium.org

Build

builder	Mac Builder
isolate_hash	630b5fe7ae1b260e78db8823309 9249b5640517b

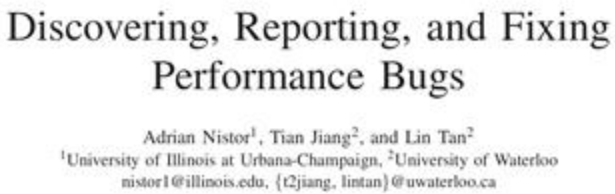
Test

task_id	3baea4beaa7f1710
bot_id	build197-b4
isolate_hash	146eb87de6d2594cc3a9ee9f351 8f69fc3d0c2c3

Values

trace	Pantip_2018-02-14_11-40-07_93865.html
trace	Pantip_2018-02-14_11-40-42_21734.html

Performance bugs are “bad” bugs



Discovering, Reporting, and Fixing Performance Bugs

Adrian Nistor[1], Tian Jiang[2], and Lin Tan[2]
[1]University of Illinois at Urbana-Champaign, [2]University of Waterloo
nistor1@illinois.edu, {t2jiang, lintan}@uwaterloo.ca

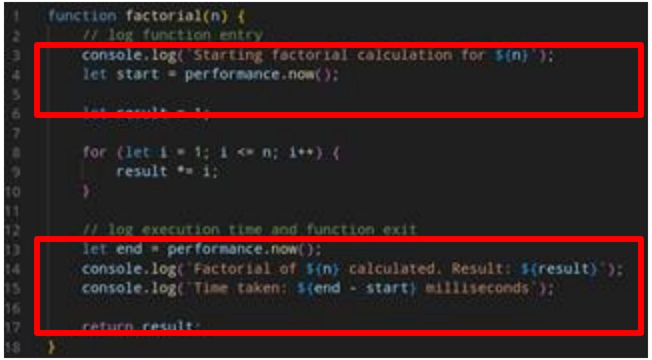
- Fixing performance bugs is usually more difficult than fixing non-performance bugs
- Performance bugs usually don’t generate incorrect results or crashes
- Difficult to diagnose:
  - system load, hardware configuration, network conditions, user-specific workflows, interactions with other systems
- Big impact on user experience

Profiling and tracing

- **Profiling** is a process to analyze and measure the performance of a program or specific parts of its code (e.g., functions).
- **Tracing** is about understanding the flow of execution and the behavior of a program.
  - Record sequential events (function calls) that occur during the execution of a program
- Both can be used to identify bottlenecks in execution time and memory

Performance analysis via instrumentation

- Embedding additional code to monitor the program's behavior
- <u>Usage:</u>
  - Source Code (**Static**): Additional instructions for data collection.
  - Binary Files (**Dynamic**): Inserting monitoring code at runtime without altering the source.
- <u>Applications:</u>
  - **Profiling**: Execution time, function call frequency, and resource usage.
  - **Tracing**: Record detailed execution flow, tracking function entries/exits and event sequences.



```
function factorial(n) {
    // log function entry
    console.log(`Starting factorial calculation for ${n}`);
    let start = performance.now();

    let result = 1;

    for (let i = 1; i <= n; i++) {
        result *= i;
    }

    // log execution time and function exit
    let end = performance.now();
    console.log(`Factorial of ${n} calculated. Result: ${result}`);
    console.log(`Time taken: ${end - start} milliseconds`);

    return result;
}
```

Flame Graphs

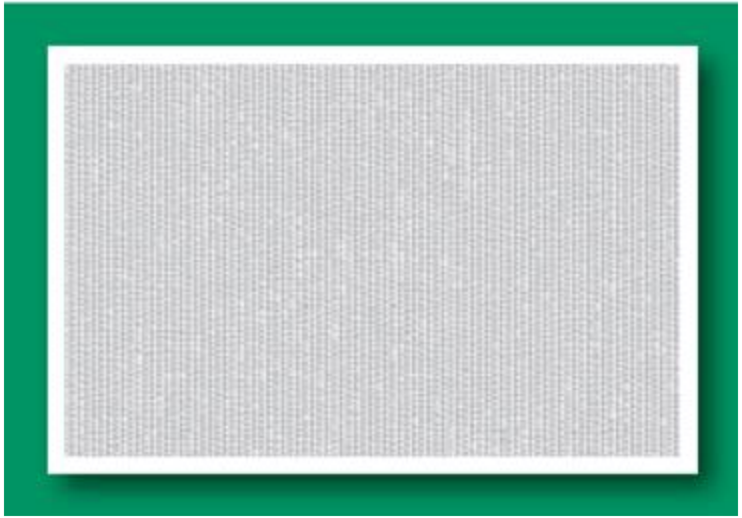
The Flame Graph

THIS VISUALIZATION OF SOFTWARE EXECUTION IS A NEW NECESSITY FOR PERFORMANCE

BRENDAN GREGG, NETFLIX

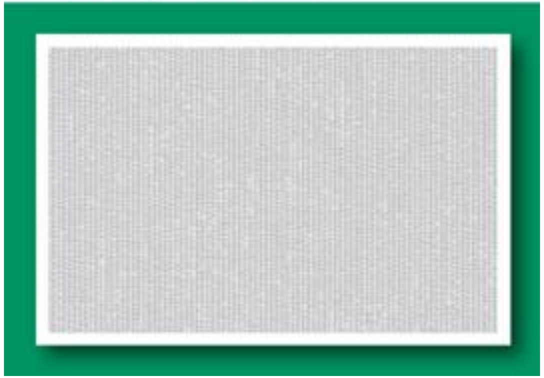
An everyday problem in our industry is understanding how software is consuming resources, particularly CPUs. What exactly is consuming how much, and how did this change since the last software version?

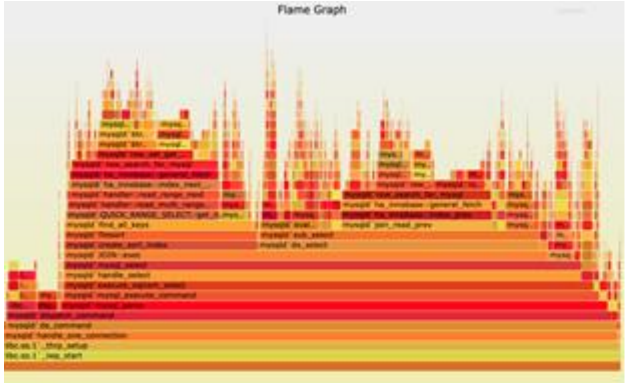
FIGURE 3: FULL MYSQL DTRACE PROFILE OUTPUT



Flame Graphs

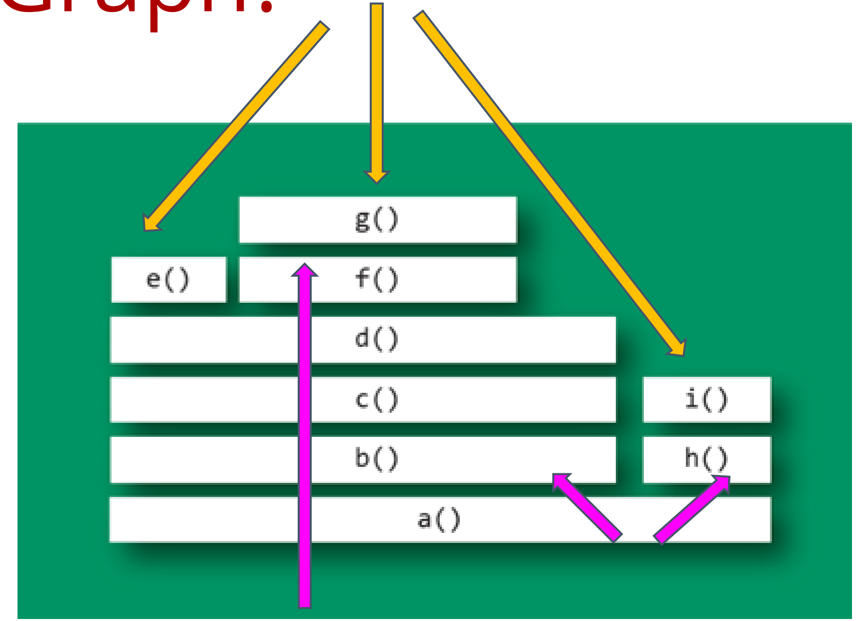
FIGURE 3: FULL MYSQL DTRACE PROFILE OUTPUT



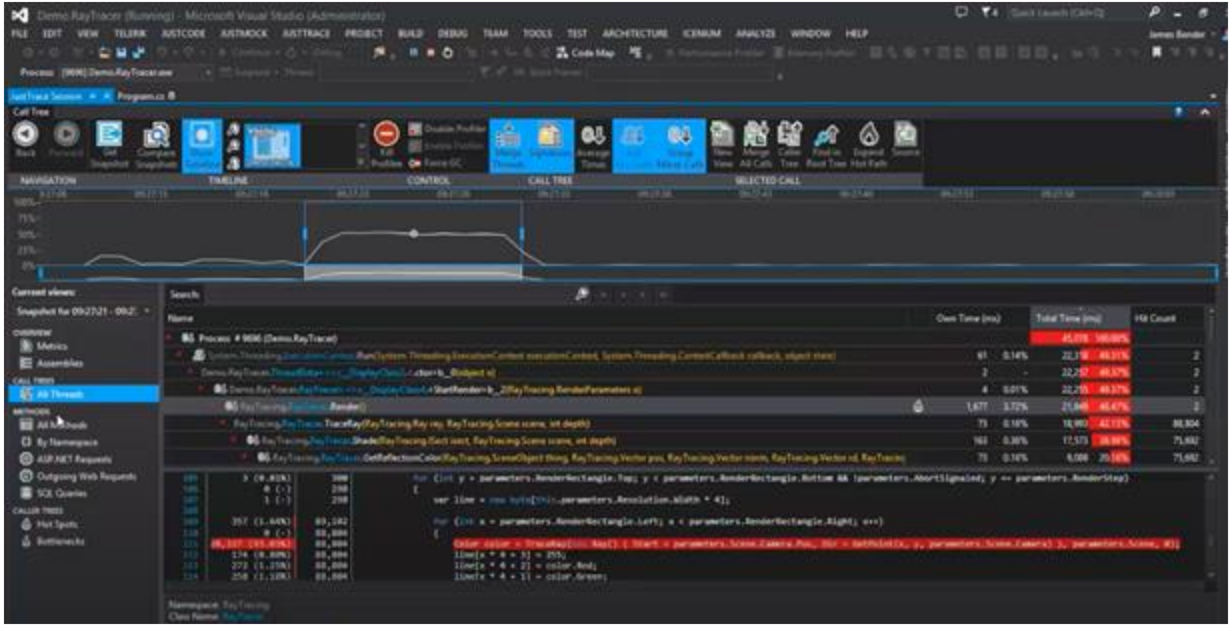


How to read a Flame Graph?

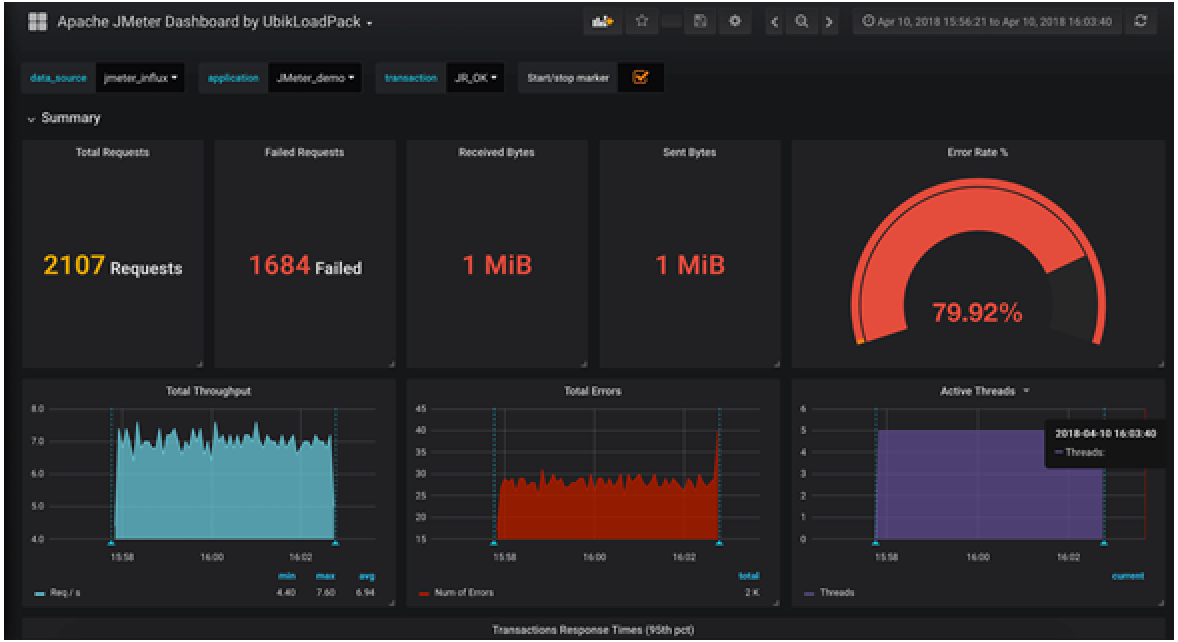
- Top edges of the flame graph show the functions that were running on when the stack trace was collected
- Top down shows ancestry
- Box width proportional to presence in stack traces



Profilers often included in IDEs


Demo.RayTracer (Running) - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW TELERIK JUSTCODE JUSTMOCK JUSTTRACE PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ICENIUM ANALYZE WINDOW HELP
Call Tree
NAVIGATION
TIMELINE
CONTROL
CALL TREE
SELECTED CALL
Current views:
All Threads
Search
Name
Own Time (ms)
Total Time (ms)
Hit Count
Process # 9696 (Demo.RayTracer)

Domain-Specific Perf Testing
(e.g. JMeter for Java web apps)



Apache JMeter Dashboard by UbikLoadPack

data_source: jmeter_influx

application: JMeter_demo

transaction: JR_OK

Start/stop marker ☑

Summary

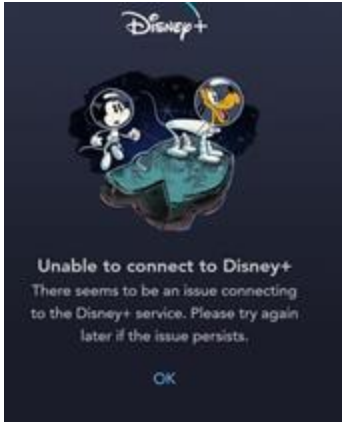
Total Requests	Failed Requests	Received Bytes	Sent Bytes	Error Rate %
2107 Requests	1684 Failed	1 MiB	1 MiB	79.92%

http://jmeter.apache.org

Stress testing

- Scalability/Robustness testing technique: test beyond the limits of normal operation.
- Can apply at any level of system granularity.
- Key idea: throw large amounts of input / requests and see how the program behaves
- Often a way to test the error-handling capabilities of the application

Real Issues: Disney+ Launch

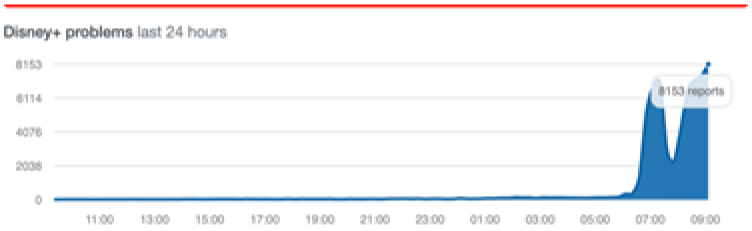


Unable to connect to Disney+

There seems to be an issue connecting to the Disney+ service. Please try again later if the issue persists.

OK

- Lots of issues reported on launch day.
- Disney had planned for a spike in traffic.
  - Tested massive concurrent video streaming capability.
- BUT: the stress was in paths other than streaming
  - User account creation
  - Logins and auth
  - Browsing old titles



Soak testing

- A system may behave exactly as expected under controlled test conditions but fail in production after extended use.
  - E.g., Memory leaks may take longer to lead to failure
- **Soak testing** a system involves applying a load over a significant period of time and observing system resilience.
- Time-consuming to run but useful to apply at big release milestones or when making infrastructure changes.

Activity:

Pick one scenario based on where you are seating

- E-Commerce Web Application (front rows)
- Automotive Software for Self-Driving Cars (middle rows)
- Mobile Gaming Application (back rows)

Discuss in groups of 2-3:

- Enumerate specific performance challenges in the your scenario.
- Pick one dynamic analysis technique to address some of these challenges.

Testing in Production

Reliability testing

- What happens when some components of a large complex system fail? Can the system recover and keep working?
- How can you test the reliability of something as complex as Netflix or Google maps or Instagram?
- One idea: simulate a large-scale deployment and induce random failures in various components

Test in Production with **Chaos Engineering**

- Another idea…

What is chaos engineering?

- "Chaos Engineering is the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production."

principlesofchaos.org

Chaos Engineering: Testing in Production

- Purposefully take down components in a **live deployment**.
- Observe system response. Do failovers work correctly?
- Tests the failure-handling and fallback capabilities of large systems.
- Useful in preparing for natural disasters or cyberattacks.

Example: Google

Terminate network in Sao Paulo for testing:

- Hidden dependency takes down links in Mexico which would have remained undiscovered without testing

Turn off data center to find that machines won’t come back:

- Ran out of DHCP leases (for IP address allocation) when a large number of machines come back online unexpectedly.

Why would you break things on purpose?

Example: Netflix

Significant deployment on AWS cloud. Hundreds of updates to microservices and infrastructure through the day.

Chaos Monkey randomly takes down AWS instances or network connections or randomly changes config files.

How to tell ”are we still good?”
Key metric: Stream Starts per Second (SPS)
Measures *availability*

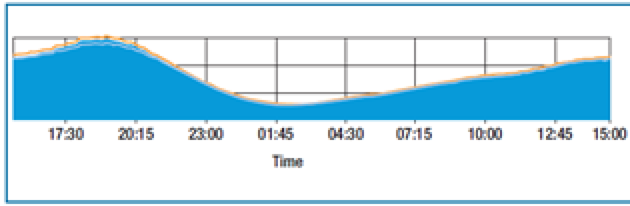


FIGURE 2. A graph of SPS ([stream] starts per second) over a 24-hour period. This metric varies slowly and predictably throughout a day. The orange line shows the trend for the prior week. The *y*-axis isn't labeled because the data is proprietary.

Testing GUIs and Usability

Automating GUI/Web Testing is Hard

- Capture and Replay Strategy
  - mouse actions
  - system events
- Test Scripts:
  - click on button labeled "Start" expect value X in field Y
- Lots of tools and frameworks
  - e.g. Selenium for browsers
- Can avoid load on GUI testing by separating model from GUI
- Beyond functional correctness?

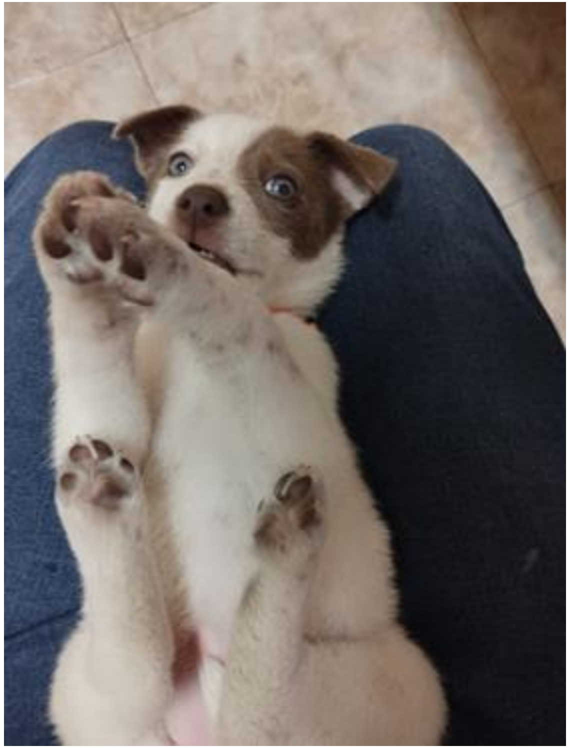
Usability: A/B testing

- Controlled randomized experiment with two variants, A and B, which are the control and treatment.
- One group of users given A (current system); another random group presented with B; outcomes compared.
- Often used in web or GUI-based applications, especially to test advertising or GUI element placement or design decisions.

Example

- A company sends an advertising email to its customer database, varying the photograph used in the ad...

Example: group A (99% of users)



Act now!

Sale ends soon!

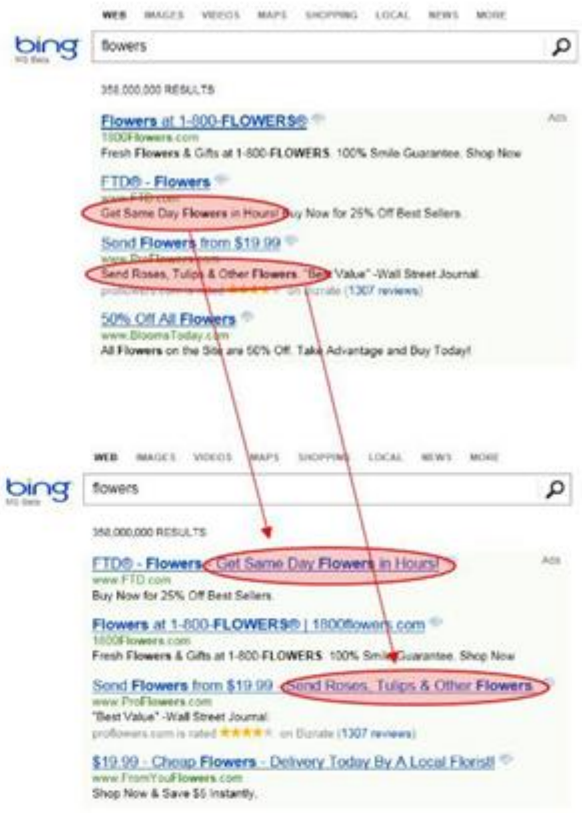
Example: group B (1%)

Act now!

Sale ends soon!

Bing Experiment

- Experiment: Ad Display at Bing
- Suggestion prioritized low
- Not implemented for 6 months
- Ran A/B test in production
- Within 2h revenue-too-high alarm triggered suggesting serious bug (e.g., double billing)
- Revenue increase by 12% - $100M annually in US
- Did not hurt user-experience metrics



WEB IMAGES VIDEOS MAPS SHOPPING LOCAL NEWS MORE

flowers

358,000,000 RESULTS

Ads

Flowers at 1-800-FLOWERS®
1800Flowers.com
Fresh Flowers & Gifts at 1-800-FLOWERS. 100% Smile Guarantee. Shop Now

FTD® - Flowers
www.FTD.com
Get Same Day Flowers in Hours! Buy Now for 25% Off Best Sellers.

Send Flowers from $19.99
www.ProFlowers.com
Send Roses, Tulips & Other Flowers. "Best Value" -Wall Street Journal.
proflowers.com is rated ★★★★ on Bizrate (1307 reviews)

50% Off All Flowers
www.BloomsToday.com
All Flowers on the Site are 50% Off. Take Advantage and Buy Today!

WEB IMAGES VIDEOS MAPS SHOPPING LOCAL NEWS MORE

flowers

358,000,000 RESULTS

Ads

FTD® - Flowers Get Same Day Flowers in Hours!
www.FTD.com
Buy Now for 25% Off Best Sellers.

Flowers at 1-800-FLOWERS® | 1800flowers.com
1800Flowers.com
Fresh Flowers & Gifts at 1-800-FLOWERS. 100% Smile Guarantee. Shop Now

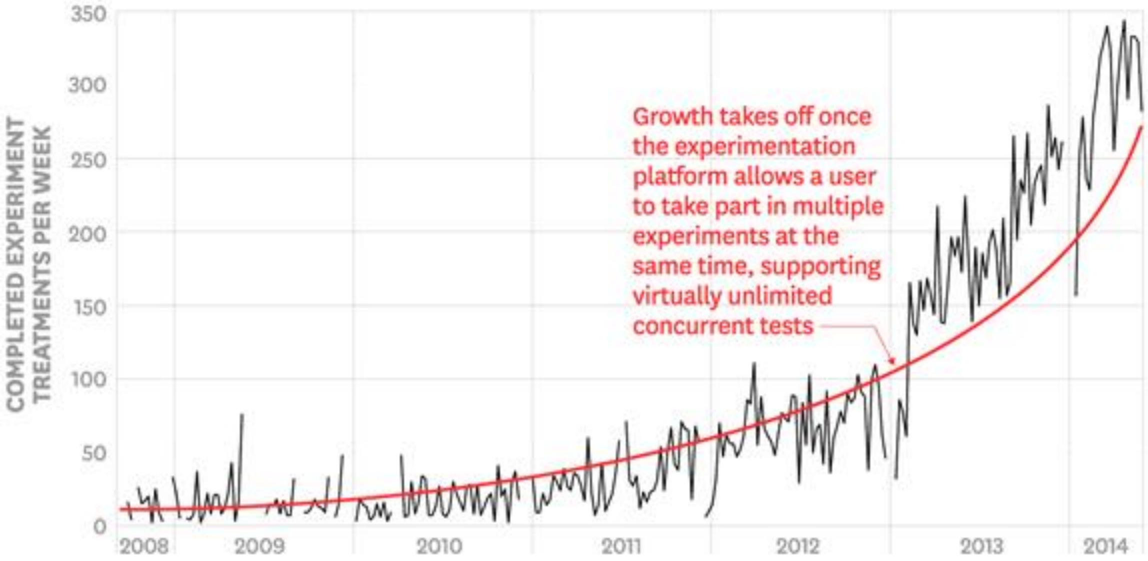
Send Flowers from $19.99 Send Roses, Tulips & Other Flowers
www.ProFlowers.com
"Best Value" -Wall Street Journal
proflowers.com is rated ★★★★ on Bizrate (1307 reviews)

$19.99 - Cheap Flowers - Delivery Today By A Local Florist!
www.FromYouFlowers.com
Shop Now & Save $5 Instantly.

Kohavi, Ron, Diane Tang, and Ya Xu. "Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing." 2020.

The power of online experimentation

The Growth of Experimentation at Bing



FROM "THE SURPRISING POWER OF ONLINE EXPERIMENTS," SEPTEMBER–OCTOBER 2017, BY RON KOHAVI AND STEFAN THOMKE

© HBR.ORG

Learning Goals

- Describe random test-input generation strategies such as fuzz testing
- Identify and discuss the key challenges associated with performance testing in software development.
- Understand the ideas behind chaos engineering and how it is used to test resiliency of cloud-based applications
- Describe A/B testing for usability
- Recommend appropriate dynamic analysis techniques for specific software quality issues.