

## Upcoming

- Literature review assignment posted, due April 11
- Project idea slides posted on Moodle for teams! (2-4 members)
- Homework 3 will be posted shortly, due April 18

1

## Reproducing Field Failures

2

## Lab Failures

When you are developing a piece of software, and you run it, use it, and it fails, what do you do to debug it?

3

## Field Failures

After you have shipped a piece of software, and a user runs it, uses it, and it fails, what can the developer do to debug it?

4

## Let's try something

Describe for me a time your software failed.

Now describe it for me as your grandpa would.

5

## Problems with Field Failures

- Users skip details
- Users describe what went wrong, not what they did
- Users aren't programmers, so they don't know what's important
- Even if the users **are** programmers, they didn't build **system** → don't know what's important

6

What's worse than a user who doesn't know what's important to report?

A user who "figured out" the system, understand exactly what the system must be doing, and is telling you his or her inferences, not observable effects.

7

How do we deal with field failures?

- We could record everything that happens at runtime, ship it back to developers.

What's wrong with this?

8

How do we deal with field failures?

- For privacy, only send stuff when something goes wrong.

What's wrong with this?

9

How do we deal with field failures?

- Anonymize inputs?
- Record sparingly?
- Deduce stuff locally?
- Find alternate inputs that lead to the same bug?

10

Let's back up

- Why worry about field failures?
  - Testing is great, but you can't catch everything
  - Software ships with bugs all the time
- Why are field failures hard to debug?
  - You don't know the circumstances
  - The environment (other installations, etc.) may play a role
  - Can't rely on the user

11

Goals

- Capture the steps necessary to replicate a bug
- Generate a test case automatically
- No effort from user

12

### There are some existing techniques RecrashJ

- Monitor a running JVM, record inputs, method invocations
- If an exception is uncaught, write down the test case that generated it
- Privacy issues, 20X overhead (sometimes), deep call stacks cause problems

13

### There are some existing techniques Scarpe

- Isolate subsystems and monitor what flows in and what flows out
- Replay exceptions, but only within a subsystem
- Faster but still 20X overhead, hasn't been evaluated very well

14

### There are some existing techniques BugRedux

- Use symbolic execution to guide test generation
- Observe an execution, record constraints that get you down a path.
- When an exception happens, figure out a different input that would follow the same path

Better for privacy, but constraint logging has to be detailed (and slow) or input reconstruction won't work + symbolic execution scales poorly

15

### Chronicler

Key idea: deterministic parts of the program are easy to recreate. It's the nondeterminism that causes many bugs.

Nondeterminism: output dependence on factors other than initial program state and input

What are some nondeterminism examples?

16

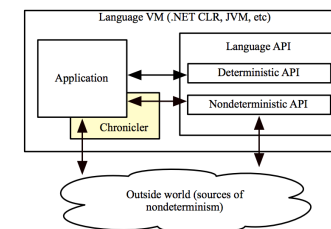
### So what kinds of things do we need to watch?

- User input (we'll call that nondeterminism)
  - file.read()
  - buf.readLine()
  - etc.
- Native calls
  - System.currentTimeMillis()
  - Random()
  - etc.

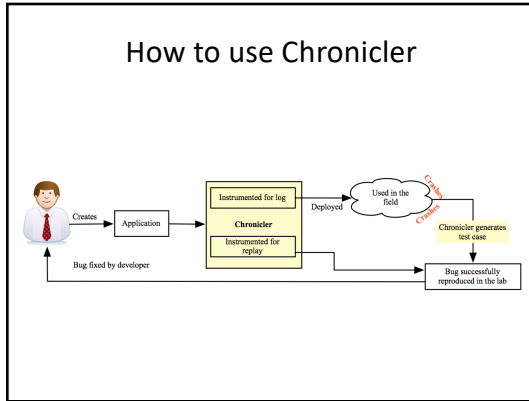
17

### How does Chronicler capture nondeterminism?

Wrap the VM and log at a higher level



18



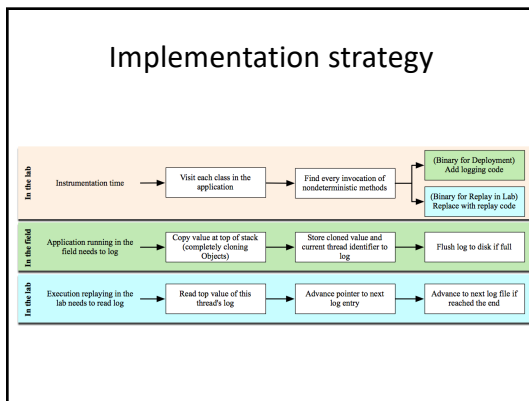
19

- ### Some implementation details
- Scan the API
    - Mark all system methods as nondeterministic
    - Mark anything that calls those as nondeterministic
    - And propagate the nondeterministic upward
  - Record and Replay
    - Instrument bytecode to record results of nondeterministic method calls
    - When replaying, simply insert recorded values
    - Can even work for GUI events (e.g., swing)

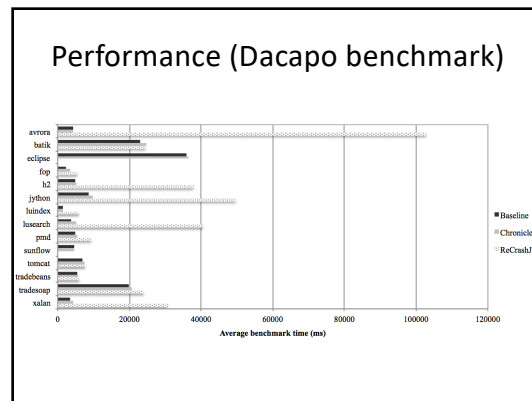
20

- ### What can this log?
- Nondeterministic event dispatching, (some) thread switches, GUI events, randomness
  - If log gets too big, flush it to a file on disk
- When do you write out a test to deliver to the developer?

21



22



23

- ### What are some Chronicer weaknesses?
- privacy is not addressed
  - some threads and processes are not recorded
  - Java can do some crazy things, like mutate its own method's parameters and use reflection to redefine a method at runtime

24

### Let's identify the 3 keys

What is the **scientific question**?

What's the key **new idea** that allows answering it?

How do you **measure** the **success** of the answer?

25

### Let's identify the 3 keys

What is the **scientific question**?

- How to replay field bugs in the lab

What's the key **new idea** that allows answering it?

How do you **measure** the **success** of the answer?

26

### Let's identify the 3 keys

What is the **scientific question**?

- How to replay field bugs in the lab

What's the key **new idea** that allows answering it?

- Recoding all nondeterminism

How do you **measure** the **success** of the answer?

27

### Let's identify the 3 keys

What is the **scientific question**?

- How to replay field bugs in the lab

What's the key **new idea** that allows answering it?

- Recoding all nondeterminism

How do you **measure** the **success** of the answer?

- Measure overhead
- Use it to find real bugs

28