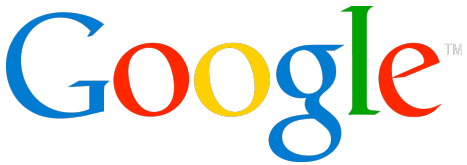


OpenJDK at Google

Jeremy Manson

Java at Google

Google

Large numbers of 0s of Java devs/code

- Programmers don't have insight into 99% of executing code
- What can OpenJDK/JVM technologies do to help?

Built a OpenJDK Team at Google

- Deploy, maintain and enhance OpenJDK/JVM
- Used by a wide variety of frontends and applications at Google.
- And many, many others!

Development at Google

Google

Single, Google-wide **giant** codebase

- Lots of zeroes of LOC, grows fast

Single test cluster

- Lots of zeroes of tests, grows fast
- All affected tests run with every checkin (mostly)

Single, very large cluster management system

- Lots of zeroes of Java servers, grows fast

One JDK to rule them all

- Best toy a PL/SE geek could want
- Lots of deployment issues...

One Slide on Deployment Issues

Google

Much of team effort is spent trying to keep up

- Java 7 caused ~20% of our tests to break
- You would be surprised at how often OpenJDK breaks at scale
- Also, trying to coordinate is a challenge

But, have some time to do fun things...

This Talk

Google

A short overview of developer workflow

- Two really long digressions
- A few really short ones

Intro

Coding

Monitoring

Conclusion

"Goo goo ga ga phbbbt code"

Coding

Coding Workflow

Google

Developers work in single SCCS

Build everything from head (mostly)

All changes have code review

- Great code review tools

Continuous testing - world explodes if your checkin breaks tests

Digression 1: Static Analysis

Google

How can we help?

We own the compiler

Static analysis

- Helps with code understanding
- Big caveat...

Coders don’t like static analysis!

Google

Lots of attempts to deploy FindBugs at Google

Coders won’t run it. Perceptions:

- Bad signal:noise ratio
- Doesn’t find important bugs
  - Lots of bugs in test / logging code
- Don’t understand bug / know of a fix
- Is outside workflow

Or so it would seem...

Let's step back

Google

Do people **really** want code like this?

```
Team team;
public Project(Team team) {
    team = checkNotNull(team);
    return this;
}
```

```
protected Source source;
@Override
public boolean equals(
    Object other){
    //  ... boilerplate
    return Objects.
       equal(source, source);
}
```

```
char tempChar = ...
if (tempChar == 0xFFFFFFFF) {
  System.err.println("Invalid
     character found while
     processing file.");
  System.exit(-1);
}
```

```
random.setSeed(supplierId +
  (string.hashCode() << 32));
```

Error Prone!

Google

You have to make it easy for them

Strategy: Make all of the warnings into compiler errors

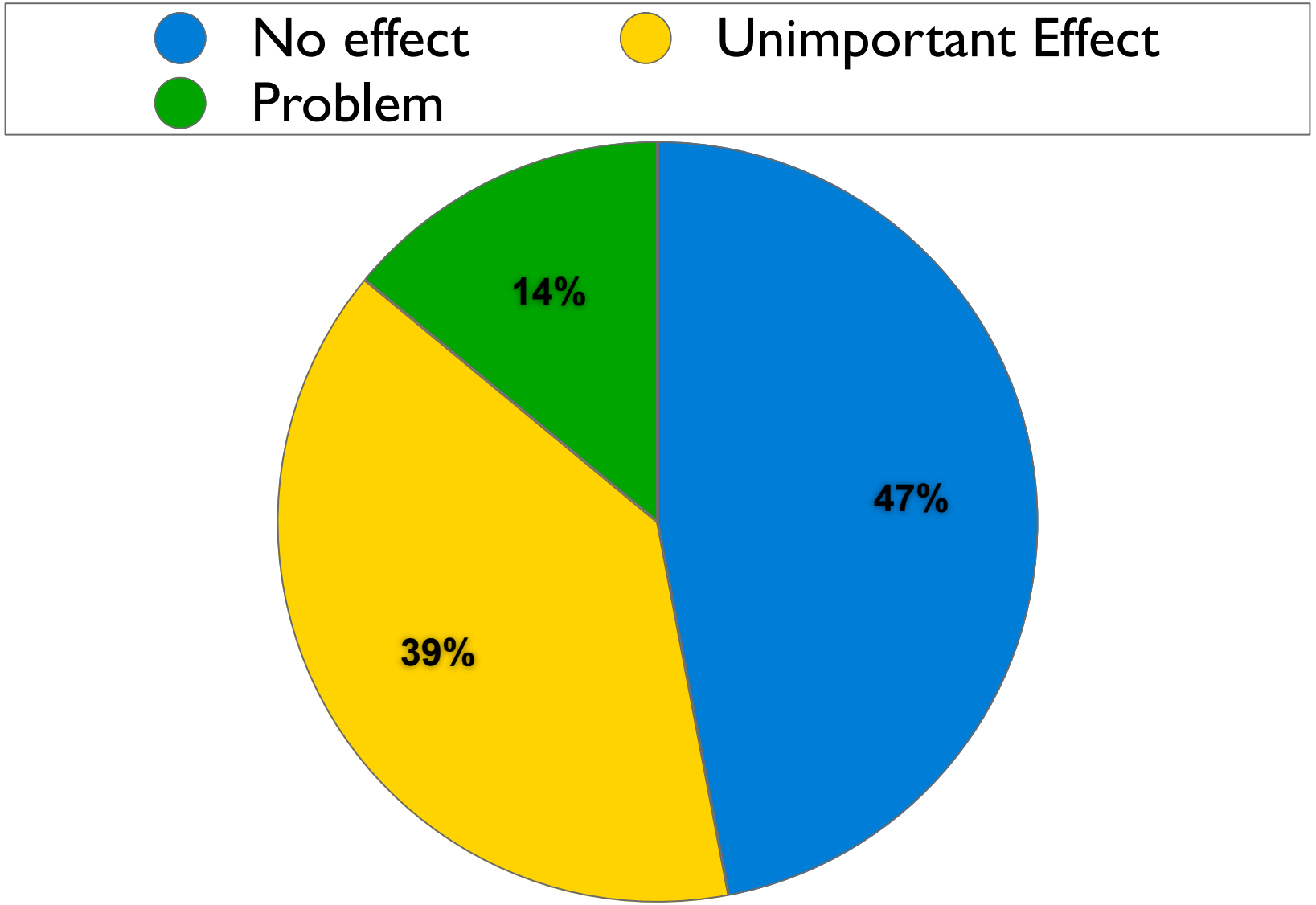
- Obviously, restricts to high value errors

Provide a suggested fix, detailed explanation

Strategy: fix **all** the bugs in Google's code base, then turn on compiler checks

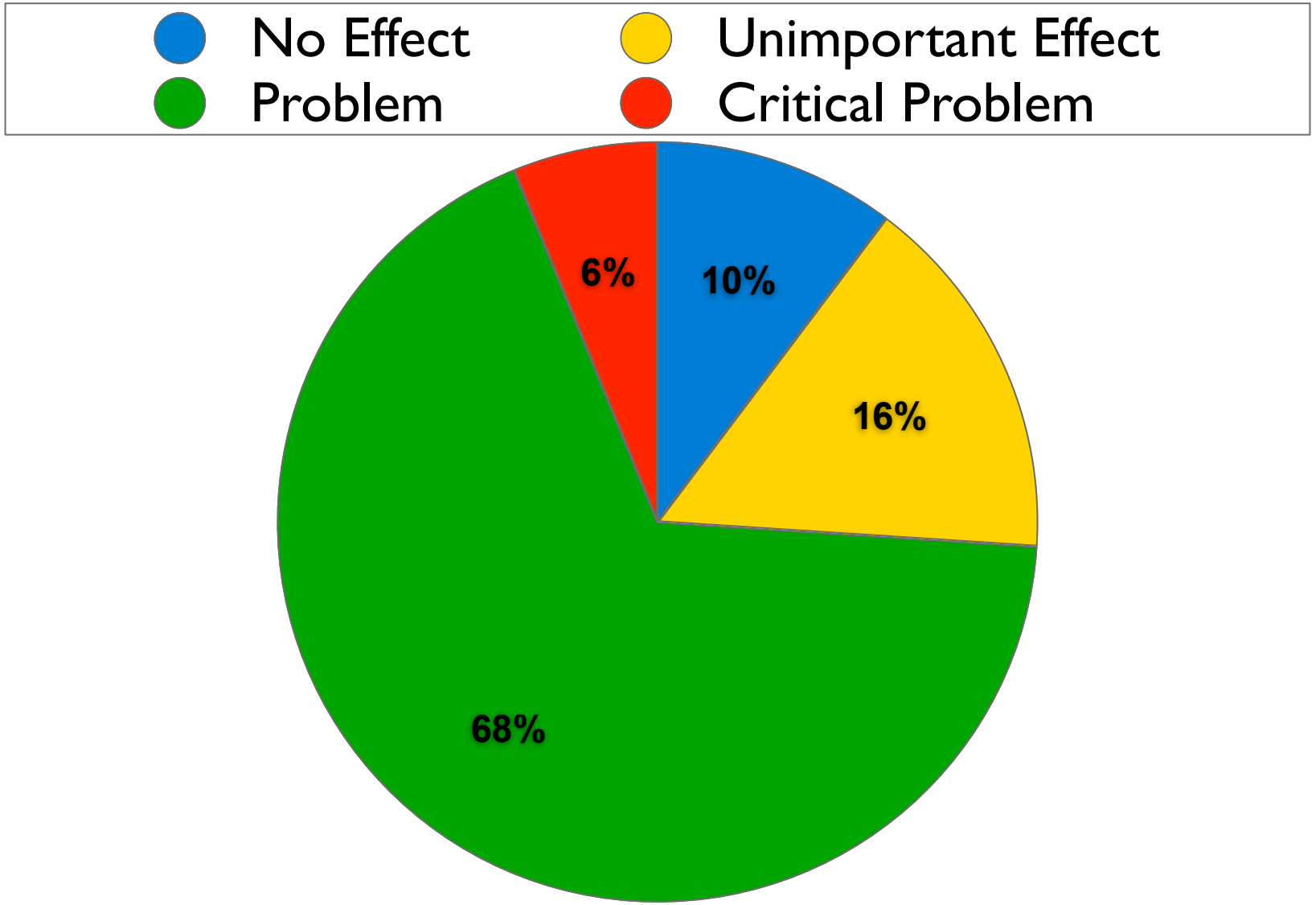
Code Review:
"Was this error likely to cause an impactful bug?"

Google



Compile time:
"Was this error likely to cause an impactful bug?"

Google



Mechanics

Google

javac has a Tree API

Have a simple, fluent API that matches patterns over the tree

- Lets people write their own (i18n, security)

Turn it into a compiler pass

Plugin framework for compiler available:

- http://code.google.com/p/error-prone

Simple API

Google

```
private static final Matcher<MethodInvocationTree> matcher = Matchers.allOf(
    methodSelect(instanceMethod(Matchers.<ExpressionTree>isArrayType(), "equals")),
    argument(0, Matchers.<ExpressionTree>isArrayType()));

/**
 * Matches calls to an equals instance method in which both the receiver and the argument are
 * of an array type.
 */
@Override
public boolean matches(MethodInvocationTree t, VisitorState state) {
  return matcher.matches(t, state);
}
```

... Still a Simple API

Google

```
public Description describe(MethodInvocationTree t, VisitorState state) {
  String receiver = ((JCFieldAccess) t.getMethodSelect()).getExpression().toString();
  String arg = t.getArguments().get(0).toString();
  SuggestedFix fix = new SuggestedFix()
      .replace(t, "Arrays.equals(" + receiver + ", " + arg + ")")
      .addImport("java.util.Arrays");
  return new Description(t, getDiagnosticMessage(), fix);
}
```

Fixing ALL the code

Google

Fix the codebase before you turn errors on

- How do you do that over N LOC?
- Can’t really run sed

We have lots of machines, and we own the compiler

Write a matcher for the AST, add it to the compiler, compile all the code

Can do **lots** of code transformations this way

Global changes

Google

Can change lots of code if you have:

- Tools that understand the code
- Well-defined transformations
- Lots of machines

Can do automated fixes, refactoring / renaming, mass deprecation...

Coders **do** like static analysis!

Google

Good signal:noise ratio

- Picking only high-value warnings now

Finds important bugs

- Lots of bugs in test / logging code
- If you are currently writing the code, you care.

Don’t understand bug / know of a fix

- Offer suggested fixes, good documentation

Is outside workflow

- Is now part of compiler

Non-Build Breaking Errors

Google

Might be expensive

- Require whole-program analysis

Might be lower-value

- Bug patterns that are sometimes correct

Working on surfacing warnings in code review

- Be interesting to compare to the value of actual reviews

Lots more work to do...

Google

Automatically generate fixes

Build some more warnings

Make Error Prone more flexible

Might be nice to standardize the Tree API / plugin framework

- Have to play catchup with every new javac revision

Testing

Dynamic Analysis

Google

Short shrift for this talk

Address sanitizer: valgrind-but-fast

Building a fast data race detector

Building “what broke my test” detector

More news as events warrant...

What then?

Google

Check in code

Automatically run tests

We do lots of stuff here to compile many-zeros of Java language code, but I'll skip the details...

Then, deployment!

Monitoring

“I can’t profile in production”

Google

Programmers have very little insight into what their code does

- They write a tiny portion of it
- They don’t write libraries
- Not responsible for deployment
- (Not responsible for other services)

Just going to talk about performance monitoring

- Some day, another talk on GC monitoring

Want always-on, no-overhead monitoring / profiling

Commercial profilers are intrusive

- Measurable overhead is unacceptable
- Tricky to aggregate results across multiple machines

Why Profiling Doesn't Work

Google

Uses bytecode rewriting / JVMTI

- High overhead
- Even sampling profilers have 10-20%

Like Heisenberg, interfering with program breeds inaccuracy

Same reasoning as μbenchmarks: JIT effects, code layout effects, etc

Why Profiling Doesn't Work

Google

Tools rely on builtin profiling for stack trace collection calls

- Profiling happens at safe points
  - Stop the world events typically used for GC
- Doesn't say what's actually running
- Doesn't account for GC time

Evaluating the Accuracy of Java Profilers, Mytkowicz et al, PLDI 2010

Had to build a profiler without these problems

JVM to the Rescue?

Google

AsyncGetCallTrace - undocumented JVMTI call

- Can be called at any time
- Don’t have to instrument code - just use system timer
- Reports time spent in GC

Extended it to report native frames

Built profiler around this

Upshot

Google

Much less overhead than traditional sampling

- 10-20% vs. basically none

Profiles running threads, not runnable ones

More accurate

Slapped together OSS proof-of-concept:

- http://code.google.com/p/lightweight-java-profiler/

What about other profiling?

Google

Heap profiling tracks 1/512K of allocations

- Tried bytecode rewriting / dtrace. **BAD.**
- Required JIT support to be fast

Call trace profiling tracks every method invocation

- <10% overhead

Experimental data race detection

- Checks for data races on 1 field at a time

Aggregation

Google

Existing profiling tools geared for single-machine apps

Need to aggregate / collate profiling results

Google-Wide Profiling

- Goes to server, gathers profile, moves on
- Does a small percentage of machines
- Collate and display results

And that's it!

Google

At this point, you start again and write some code.

Got a few more minutes...

Garbage Collection

“GC Eats Up My CPU”

Google

GC is a large percentage of Java CPU time

- Allocation is cheap, GC isn’t
- Is it better to GC for 2 minutes or to restart your server in 30 seconds?

To date: lots of incremental improvements

In short: Make the GC Better

Google

Focused on CMS

- G1 wasn't ready for prime time

Balancing #threads to #cores

- Improvements of up to 30% in YG GC

Parallelizing full GC for CMS

- Improvements of 2-4x on full GC time

Parallelizing initial mark / remark in CMS

- Decrease those pauses 2-4x

In short: Make the GC Better

Google

Partially defrag heap during CMS phases

- Decrease full GC up to 90% (or eliminate completely)

Give back unused RAM to the system

- 20-30% RAM savings in our servers

Experimental Stack Allocation

Google

JIT-based / escape analysis is nice, but has limitations

Just let devs allocate on the stack?

Playing with an experimental manual stack allocator

So far, good results on benchmarks

- That's what they said about escape analysis
- Project on hold until someone to work on it

Obviously, this is well worth our time...

Google

By the way, did I mention we're hiring?

Conclusions

Operating Servers at Scale

Google

Static analysis needs to be **in your face**

- People love compilers, right?

Profiling needs to be low-overhead, accurate, scalable

- Profilers are slow and inaccurate

Garbage collection is a constant struggle