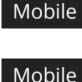


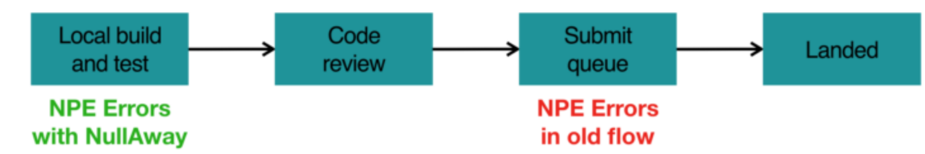
Mobile

Mobile

Engineering NullAway, Uber's Open Source Tool for Detecting NullPointerExceptions on Android

Manu Sridharan October 19, 2017

0



3

160 SHARES

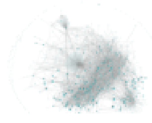
Sign up for Uber Engineering updates:

Your email address

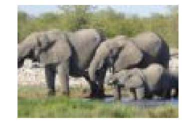
Subscribe

Popular Articles

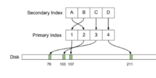
Meet Michelangelo: Uber's Machine Learning Platform
September 5, 2017



Introducing Domain-Oriented Microservice Architecture
July 23, 2020



Uber's Big Data Platform: 100+ Petabytes with Minute Latency
October 17, 2018



Why Uber Engineering Switched from Postgres to MySQL

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

external data, static analysis tools play a key role in ensuring high code reliability by detecting potential bugs before updates are shipped to users.

Previously, Uber deployed third-party static analysis tools to detect potential NullPointerExceptions (NPE), a leading cause of app crashes, and maintain reliability in our Android codebase. However, as our codebase grew, we found that these tools did not quite meet our needs for providing strong checks and quick feedback to engineers.

To address this, Uber developed NullAway, a fast and practical tool to help eliminate NPEs. NullAway significantly improved our developers' productivity, while maintaining the strong checking we required to deploy safely. And now, we have contributed this tool to the open source community so others can build more reliable apps, too!

In this article, we discuss our motivation behind developing NullAway, outline how we built the tool, and provide instructions on how to use it for your own Android apps and Java projects.

June 27, 2018

Introducing Ludwig, a Code-Free Deep Learning Toolbox
February 11, 2019

The Uber Engineering Tech Stack, Part I: The Foundation
July 19, 2016

Introducing AresDB: Uber's GPU-Powered Open Source, Real-time Analytics Engine
January 29, 2019

Forecasting at Uber: An Introduction
September 6, 2018

Building Reliable Reprocessing and Dead Letter Queues with Apache Kafka
February 16, 2018

Categories

AI
Announcement

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

General Engineering
Mobile
Open Source
Team Profile
Uber Data

Mobile crashes can cause significant problems for our users, such as preventing riders from requesting a trip in a timely manner or drivers from accepting rides. NPEs, which occur when a null pointer is dereferenced in Java, are a frequent cause of crashes in Android apps. Our strategy at Uber has been to use static code analysis tools to prevent NPE crashes as much as possible. In 2016, Uber deployed the Facebook Infer and Eradicate tools for static detection of potential NPEs. Alongside our Runtime Annotation Validation Engine (RAVE) validation, these tools reduced the number of NPEs observed in our apps in production by an order of magnitude.

To maximize reliability, we wanted to guarantee that no code could be merged into our apps until all possible NPE warnings were fixed; this way, our master branch would always be in a “green” state (meaning the build has passed all relevant tests) with zero warnings. Obtaining this guarantee required running these tools on our Submit Queue, the final stage of our continuous integration pipeline, such that any error from the tools would block new code from merging.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

1. **Late feedback:** For diffs with NPE problems, developers only received warnings at the final stage of making a code change. This late feedback could lead to a frustrating experience: a developer's change could pass code review and all other checks, only to be rejected on the Submit Queue due to an easy-to-fix issue. The developer may well have moved on to another task by this point, and would have to context-switch to return to the code and address the warning.
2. **Higher latency, decreased developer productivity:** The overall latency of the Submit Queue experience increased, thereby reducing developer productivity. Since many diffs failed on the Submit Queue with NPE warnings, they needed to be submitted multiple times, increasing overall queue lengths.

Figure 1: There are four stages of Uber Engineering's continuous integration pipeline on Android. Before NullAway, Uber developers would only notice NPE

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

running them earlier in the process, so we decided to build a new solution that would be fast enough to run with low latency at every stage of our development pipeline, even during local builds. And an added benefit of developing our own fix is that we could save on machine resources as we continue to grow our codebase.

Our answer? NullAway.

Introducing NullAway

At its core, NullAway is an open source type-based NPE checker for Java code. To use NullAway, you must first add @Nullable annotations in your code wherever a field, method parameter, or return value may be null. (We already had these annotations in our codebase due to our previous use of Eradicate.) Given these annotations, NullAway performs a series of local consistency checks to ensure that any pointer that gets dereferenced in your code cannot be null.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

reuse much of the work already done by the compiler, such as code parsing and type checking. Further, NullAway and Error Prone integrate directly into the fast, in-memory parallel builds supported by Buck, the build tool we use for our Android code. Therefore, NullAway can run much more quickly than tools that run outside the normal build process.

We found that NullAway added only a small overhead to normal build times (around 10 percent in our measurements). As a result, rather than only running on the Submit Queue, we configured NullAway to run on every single build of our Android code.

The value of integrating NullAway into all of our Java builds is threefold:

- **Immediate feedback:** Integration into all builds enables developers to get immediate feedback whenever they introduce a potential NPE, rather than having to wait for the Submit Queue.
- **No NPE checker:** The build integration means we no longer need to run a nullness checker as a separate job on the Submit Queue, saving significant machine resources.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

even more pronounced after our move to a monorepo.

Using NullAway

Figure 2: In three easy steps, NullAway determines if an expression "e" in method "m" can be null.

To get an idea of how NullAway works, let us consider how it determines whether some expression in the program can be null,

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
class A {
 @Nullable Object f;
}
static void m(A x) {
 if (x.f != null) {
   System.out.println(x.f.toString());
 }
}
```

In the example above, NullAway tries to show that x is not null, to ensure that `x.f` does not cause an NPE, and also that `x.f` is not null at the call `x.f.toString()`. For expressions that should not be null, NullAway first tries to quickly prove that the expression is obviously non-null, e.g., by checking if its type is not `@Nullable` or if it is an expression like `new Object()` that cannot be null. For the example, since the declaration of `x` is not annotated `@Nullable`, NullAway can assume it is not null, thereby showing `x.f` is safe. NullAway enforces this assumption by checking that a `@Nullable` expression is never passed as a parameter to method `m`.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

existing null checks in the code. For the `x.f.toString()` call in above example, quick checks fail to show that `x.f` cannot be null, since the `A.f` field is `@Nullable`. However, our data-flow analysis uses the enclosing conditional check that `x.f != null` to show that the `x.f.toString()` call is safe. Since the data-flow analysis can be expensive (it requires computing a control-flow graph and running a fixed-point computation), NullAway runs the analysis only once per method and caches the result.

On the NullAway GitHub page, there are detailed instructions for how to run the tool on your Android app or other Java code. For more details on NullAway's checks, error messages, and limitations, see our detailed implementation guide.

Next steps

With NullAway, we are able to maintain strong static checking for NPEs by other tools while also improving developer productivity. We hope you find NullAway useful for improving the reliability of your Java codebase.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Manu Sridharan is a software engineer on Uber's Mobile Developer Platform Team, leading efforts to apply static analysis across Uber's codebase. Earlier this year, he delivered a tech talk at Curry On about how static analysis and software architecture are leveraged to improve application quality at Uber.

Comments

0 Comments Sort by **Oldest**

Add a comment...

Facebook Comments Plugin

3

160 SHARES

TAGS Android Buck Checker Framework data-flow analysis Development Eradicate Error Prone Feature Flags Infer Java Mobile mobile dev

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Previous article

Meet Horovod: Uber's Open Source Distributed Deep Learning Framework for TensorFlow

Next article

Scaling Reliable Transportation for India: Meet Uber Bangalore Engineering

Manu Sridharan

Manu Sridharan, formerly a staff engineer at Uber, is an Associate Professor of Computer Science and Engineering at the University of California, Riverside.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Uber Engineering

ubereng@uber.com
@ubereng
UberEngineering
Uber Engineering
UberEngineering
UberEngineering

AI
Architecture
Culture
General Engineering
Mobile
Open Source
Uber Data

Uber Open Source
Uber Research
Uber.com
Uber Eats
Uber for Business
Help
Newsroom
Careers

© 2021 Uber Technologies Inc.

Privacy Policy

Terms and Conditions

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD