Mutation Testing

Leaving the Stone Age

2017

whoami

- iOS Developer by day
- compiler hacker by night
- https://twitter.com/1101_debian
- https://lowlevelbits.org
- https://systemundertest.org

Outline

- Quality of Software
- Unit Testing
- Mutation Testing
- Mull
- Showcase: LLVM Test Suite

Quality of Software

Quality of Software

- Formal Verification

Quality of Software

- Formal Verification
- Fuzz Testing

Quality of Software

- Formal Verification
- Fuzz Testing
- Unit Testing + Code Coverage

Quality of Software

- Formal Verification
- Fuzz Testing
- **Unit Testing + Code Coverage**

Unit Testing

Unit Testing

```
int sum(int a, int b) {
  return a + b;
}
```

Unit Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

Unit Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

Failed Tests: 0

Passed Tests: 1

Code Coverage: 100%

Mutation Testing

```
run_test(program, test)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
result = run_test(mutant, test)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
result = run_test(mutant, test)
if (result == Failed)
  report_killed_mutant(mutant, test)
```

Mutation Testing

```
run_test(program, test)
mutant = mutate(program)
result = run_test(mutant, test)
if (result == Failed)
  report_killed_mutant(mutant, test)
else
  report_survived_mutant(mutant, test)
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

```
int sum'(int a, int b) {
  return a * b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

```
int sum'(int a, int b) {
  return a * b;
}
```

```
int sum''(int a, int b) {
  return a - b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

test **passed** ->
  mutant **survived**

```
int sum'(int a, int b) {
  return a * b;
}
```

```
int sum''(int a, int b) {
  return a - b;
}
```

Mutation Testing

```
void test() {
  assert(sum(5, 10) > 0);
}
```

```
int sum(int a, int b) {
  return a + b;
}
```

```
int sum'(int a, int b) {
  return a * b;
}
```

```
int sum''(int a, int b) {
  return a - b;
}
```

test **passed** ->
  mutant **survived**

test **failed** ->
  mutant **killed**

Mutation Testing

Total Mutants: 2

Killed Mutants: 1

Survived Mutants: 1

Mutation Score = killed / total * 100%

Mutation Score: **50%**

Mutation Testing

- First proposed by Richard Lipton in **1971**

Mutation Testing

- First proposed by Richard Lipton in **1971**
- First implemented by Timothy Budd in **1980**

Mutation Testing

- First proposed by Richard Lipton in **1971**
- First implemented by Timothy Budd in **1980**
- Studies say that MT was able to detect **70%-90%** of real faults

Mutation Testing

- Generates lots of data

Mutation Testing

- Generates lots of data
- Time consuming

Mutation Testing

- Generates lots of data
- Time consuming
- Languages are not mutation-testing-friendly

Mutation Testing

- Generates lots of data
- Time consuming
- Languages are not mutation-testing-friendly
- Problem of a Human Test Oracle

Mutation Testing

- Generates lots of data
- Time consuming
- Languages are not mutation-testing-friendly
- Problem of a Human Test Oracle
- "Excuse me, but I write good tests"

Mull

Mull

- Smart mutant selection

Mull

- Smart mutant selection
- Control over data generation

Mull

- Smart mutant selection
- Control over data generation
- Runtime compilation

Mull

- Smart mutant selection
- Control over data generation
- Runtime compilation
- Operates on LLVM IR level

Mull

- Smart mutant selection
- Control over data generation
- Runtime compilation
- Operates on LLVM IR level
- Language agnostic*

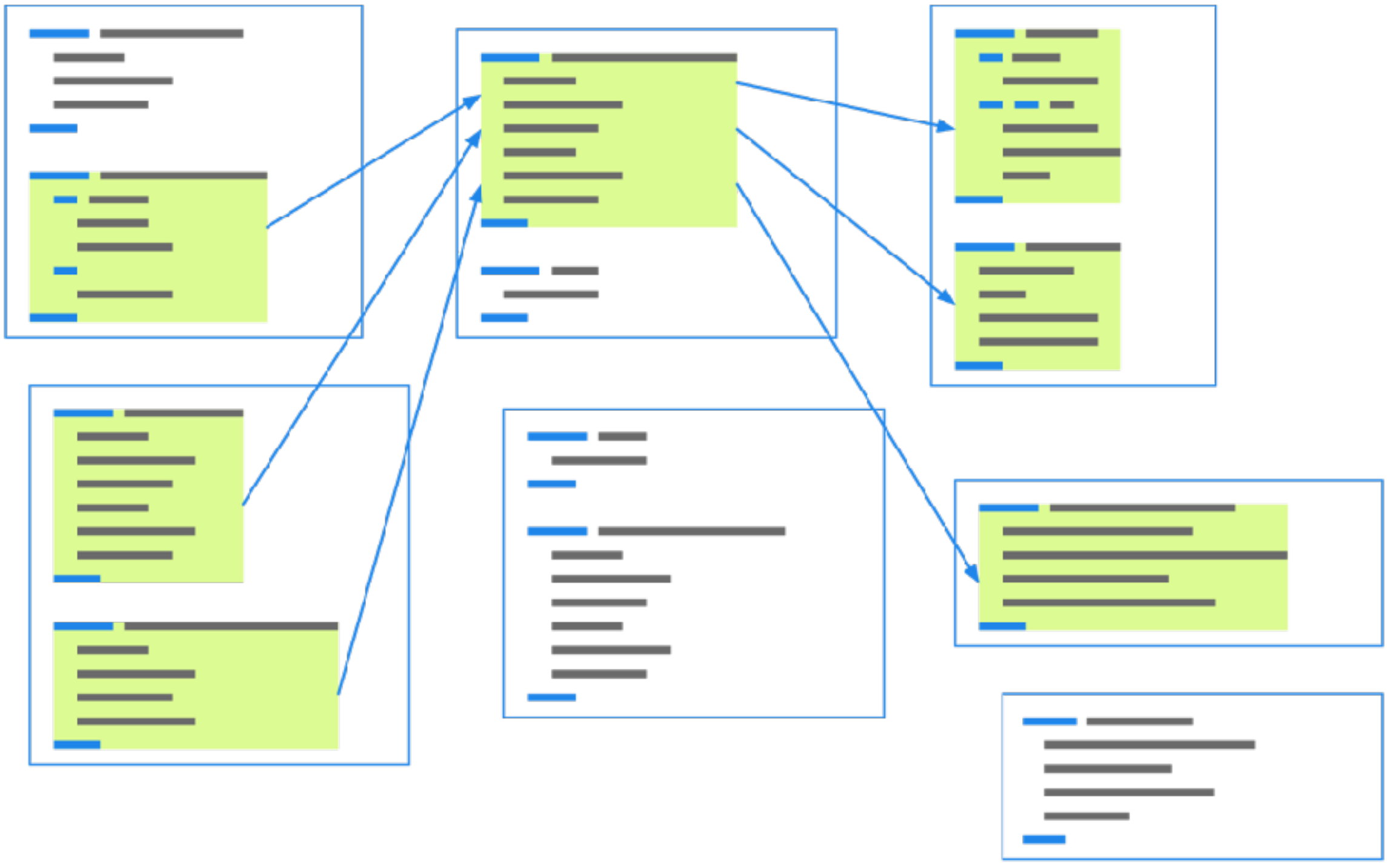
Mutant Selection

Mutant Selection

Mutant Selection

Mutant Selection

Mutant Selection



Mutant Selection

IRTests: 238 tests

Before:

391 modules

85 minutes

IRTests: 238 tests

Before:	After:
391 modules	124 modules
85 minutes	48 minutes

Mutation Control

Mutation Control

IRTests: 238 tests

Distance: 2

Number of mutants: ~1.5k

Real execution time: ~1 hour

IRTests: 238 tests

Distance: 2

Number of mutants: ~1.5k

Real execution time: ~1 hour

Distance: 29

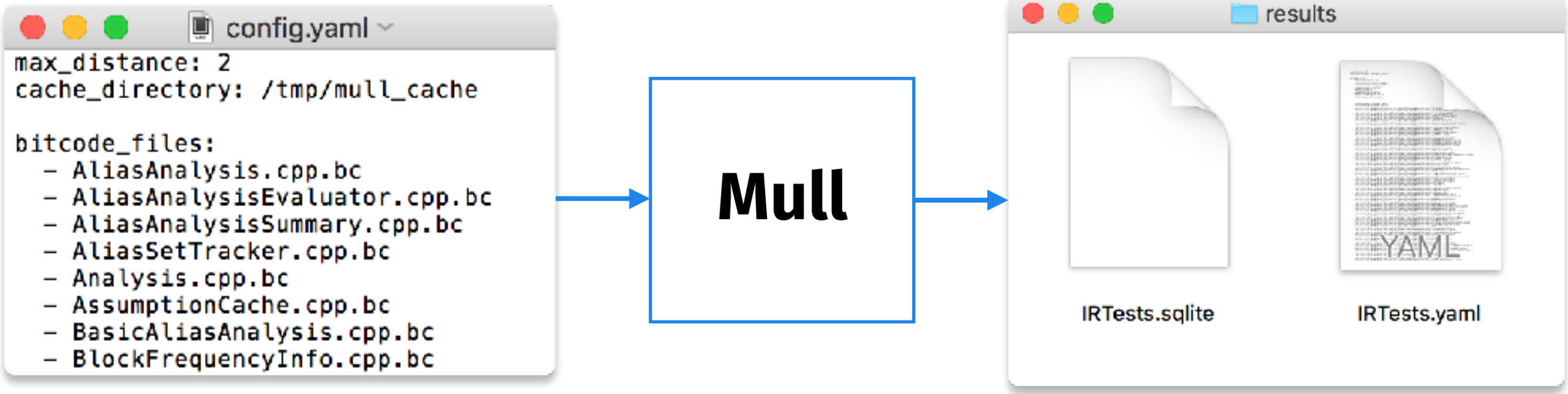
Number of mutants: ~18k

Approximate execution time: ~11 days

Mutant Control

System Design

System Design



System Design

Core

- Driver
- Reporter
- Mutation Operators

System Design

Core

- Driver
- Reporter
- Mutation Operators

Toolchain

- JIT Compiler
- Object Cache

System Design

Core	Toolchain	Test Framework
• Driver • Reporter • Mutation Operators	• JIT Compiler • Object Cache	• Test Finder • Test Runner

System Design

Core

- Driver
- Reporter
- Mutation Operators

Toolchain

- JIT Compiler
- Object Cache

Test Framework

Google Test

- Test Finder
- Test Runner

System Design

Core

- Driver
- Reporter
- Mutation Operators

Toolchain

- JIT Compiler
- Object Cache

Test Framework

Google Test

XCTest

- Test Finder
- Test Runner

Showcase

- IRTests
- ADTTests

IRTests

Number of tests: 238

Number of mutants: 1.5k

Mutation score: 43%

https://lowlevelbits.org/IRTests/

ADTTests

Number of tests: 465

Number of mutants: 1.5k

Mutation score: 66%

http://lowlevelbits.org/ADTTests/

TripleTests.*

Number of tests:	13
Number of mutants:	624
Mutation score:	83%

TripleTests.*

```
T.setArch(Triple::mips64);
EXPECT_EQ(Triple::mips64el,
          T.getLittleEndianArchVariant().getArch());
```

Triple_82_4_0_remove_void_function_mutation_operator 0/1

Affected Tests:

```
TripleTest.EndianArchVariants
```

/usr/local/LLVM/llvm/lib/Support/Triple.cpp:1413

```
  case Triple::tce:         T.setArch(Triple::tcele);     break;
                              ^
```

Survived

Distance: 1

Duration: 366ms

Caller path:

```
/usr/local/LLVM/llvm/unittests/ADT/TripleTest.cpp:693
  /usr/local/LLVM/llvm/lib/Support/Triple.cpp:1413
```

Caller path (source code):

```
  case Triple::tce:         T.setArch(Triple::tcele);     break;
```

stdout:

```
Note: Google Test filter = TripleTest.EndianArchVariants
```

TripleTests.*

```
T.setArch(Triple::mips64);
EXPECT_EQ(Triple::mips64el,
          T.getLittleEndianArchVariant().getArch());

T.setArch(Triple::tce);
EXPECT_EQ(Triple::tcele,
          T.getLittleEndianArchVariant().getArch());
```

r294095, r294096

TripleTests.*

```
Triple T = Triple("");
T.setObjectFormat(Triple::ELF);
EXPECT_EQ(Triple::ELF, T.getObjectFormat());
```

TripleTests.*

```
Triple T = Triple("");
// T.setObjectFormat(Triple::ELF);
EXPECT_EQ(Triple::ELF, T.getObjectFormat());
```

TripleTests.*

```
Triple T = Triple("");
T.setObjectFormat(Triple::ELF);
EXPECT_EQ(Triple::ELF, T.getObjectFormat());

T.setObjectFormat(Triple::MachO);
EXPECT_EQ(Triple::MachO, T.getObjectFormat());
```

r294104

IRTests

```
if (foobar) {
  fastVersion();
} else {
  slowVersion();
}
```

Open Questions

Open Questions

- Integration

Open Questions

- Integration
- UX

Open Questions

- Integration
- UX
- Next Language

Open Questions

- Integration
- UX
- Next Language
- Many unknowns

Project: https://github.com/mull-project/mull

Contact: alex@lowlevelbits.org

Updates: https://twitter.com/1101_debian

Questions?

Project: https://github.com/mull-project/mull

Contact: alex@lowlevelbits.org

Updates: https://twitter.com/1101_debian