# Reasonable Scala Compiler

**Eugene Burmako**
**Twitter, Inc.**

**11/16/2017**

# Agenda

- Vision
- Status
- Architecture

# Credits

- Martin Odersky
- Grzegorz Kossakowski
- Denys Shabalin
- David Keenan
- Shane Delmore
- My other colleagues at Twitter

# Vision

# Mission statement

An experimental Scala compiler focused on compilation speed

# Research goal

5-10x compilation speedup for typical Scala codebases

# Axiom #1: Rewrite from first principles

- Existing Scala compilers are too big for radical experiments
- Scala 2.12.4: ~140kloc of Scala code in src/reflect and src/compiler
- Dotty 0.4.0-RC1: ~90kloc of Scala code in compiler/src

# Axiom #2: Start small

- Starting small provides a tractable baseline
- Thanks to Scalafix, even a small subset of the language is useful
- The current iteration of Rsc was developed in less than a month

# Axiom #3: Contribute to the community

- We do not intend to compete with Scalac or Dotty
- We will be performing experiments and sharing our findings, so that other compilers can potentially adopt our best ideas

# Status

# Implementation

- Language model: ~1.5kloc
- Tokenizer: ~0.5kloc
- Parser: ~2.0kloc
- Prototype typechecker: ~1.5kloc
- Codegen: out of scope

# Case study

- An implementation of RE2 ported from Java
- Unique opportunity for direct comparisons with Java compilation speed
- ~11kloc of nontrivial but simple enough Scala code

# Disclaimer

- Rsc only supports a small subset of Scala
- Rsc only loads stubs, doesn’t fully read Scala signatures
- Rsc only resolves names, doesn’t fully typecheck code
- Details of benchmarking methodology explained in the documentation

# #1: Hot Rsc is >20x faster than hot Scalac

| | Cold | Hot |
|---|---|---|
| RscNativeTypecheck | 300.146 ms | 282.468 ms |
| RscTypecheck | 476.843 ms | 33.685 ms |
| ScalacTypecheck | 4326.812 ms | 712.872 ms |
| ScalacCompile | 8098.704 ms | 1691.732 ms |
| JavacCompile | 851.787 ms | 76.164 ms |

# #2: Cold Rsc is >10x faster than cold Scalac

| | Cold | Hot |
|---|---|---|
| RscNativeTypecheck | 300.146 ms | 282.468 ms |
| RscTypecheck | 476.843 ms | 33.685 ms |
| ScalacTypecheck | 4326.812 ms | 712.872 ms |
| ScalacCompile | 8098.704 ms | 1691.732 ms |
| JavacCompile | 851.787 ms | 76.164 ms |

# #3: Rsc looks comparable with Javac

| | Cold | Hot |
|---|---|---|
| RscNativeTypecheck | 300.146 ms | 282.468 ms |
| RscTypecheck | 476.843 ms | 33.685 ms |
| ScalacTypecheck | 4326.812 ms | 712.872 ms |
| ScalacCompile | 8098.704 ms | 1691.732 ms |
| JavacCompile | 851.787 ms | 76.164 ms |

# Architecture

# Example (1)

```
class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
}
```

```
class B {
  def c: C = {
    ...
  }
}
```

# Example (2)

```
class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
  def b: B = { ... }
}
```

```
class B {
  def c: C = {
    ...
  }
  def a: A = { ... }
}
```

# Example (3)

```
import M._

class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
  def b: B = { ... }
}
```

```
object M extends X {
  class B {
    def c: C = {
      ...
    }
    def a: A = { ... }
  }
}
```

# Typechecking in Scalac/Dotty

Namer (lazily computes signatures) + typer:

- Laziness significantly simplifies the implementation
- But: makes it hard to build a mental performance model of typechecking
- But: makes it VERY hard to parallelize typechecking

# Typechecking in Rsc

Single eager pass over the code, split into four parts:

- Schedule
- Scope
- Outline (potentially parallel)
- Typecheck (potentially parallel)

# Example (parse)

```
import M._

class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
  def b: B = { ... }
}
```

```
object M extends X {
  class B {
    def c: C = {
      ...
    }
    def a: A = { ... }
  }
}
```

# Example (schedule)

```
import M._

class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
  def b: B = { ... }
}
```

```
object M extends X {
  class B {
    def c: C = {
      ...
    }
    def a: A = { ... }
  }
}
```

# Example (scope)

```
import M._

class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
  def b: B = { ... }
}
```

```
object M extends X {
  class B {
    def c: C = {
      ...
    }
    def a: A = { ... }
  }
}
```

# Example (outline)

```
import M._

class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
  def b: B = { ... }
}
```

```
object M extends X {
  class B {
    def c: C = {
      ...
    }
    def a: A = { ... }
  }
}
```

# Example (typecheck)

```
import M._

class A {
  def foo: Foo = {
    val b: B = ...
    b.c
  }
  def b: B = { ... }
}
```

```
object M extends X {
  class B {
    def c: C = {
      ...
    }
    def a: A = { ... }
  }
}
```

# Summary

# Summary

- We have a very fast prototype typechecker for a small subset of Scala
- Check it out at https://github.com/twitter/reasonable-scala