FastParse

Fast, Modern, Object-Oriented Parser Combinators

Li Haoyi, Parsing@SLE 24 Oct 2015

Who Am I

Li Haoyi

Dropbox Dev-Tools, previously Web-Infra

Worked on Scala.js, Ammonite Scala REPL in free time

What is Fastparse?

FastParse

```
> import fastparse.all._

> val ab = P( "a".rep.! ~ "b" ~ End )

> ab.parse("aaaaaaab")
Success(aaaaaaa,8)

> ab.parse("aaaaaaac")
Failure("b":7 ..."c")
```

A Recursive Descent Parser Combinator library

```
"hello" : P[Unit]

a.!     : P[String] // Capture

a ~ b   : P[(A, B)]

a | b   : P[T >: A >: B]

a ~! b  : P[(A, B)] // Cut

a.rep() : P[Seq[A]]

a.?     : P[Option[A]]

!(a), &(a) // Pos/Neg Lookahead
```

```
a.map(f: A => B): P[B]

a.flatMap(f: A => P[B]): P[B]

a.filter(f: A => Boolean): P[A]

a.log(s: String): P[A]

CharPred(f: Char => Boolean)

CharIn(s: Seq[Char]*)

CharsWhile(f: Char => Boolean, min: Int = 1)

StringIn(strings: String*)
```

Live Demo

JSON-lite

FastParse is...

A new Parser Combinator library for Scala

Very convenient (in code, no special build step)

Great error reporting

Bog-standard recursive-descent/PEG

“bat-out-of-hell fast” - Mark Waks

Super flexible

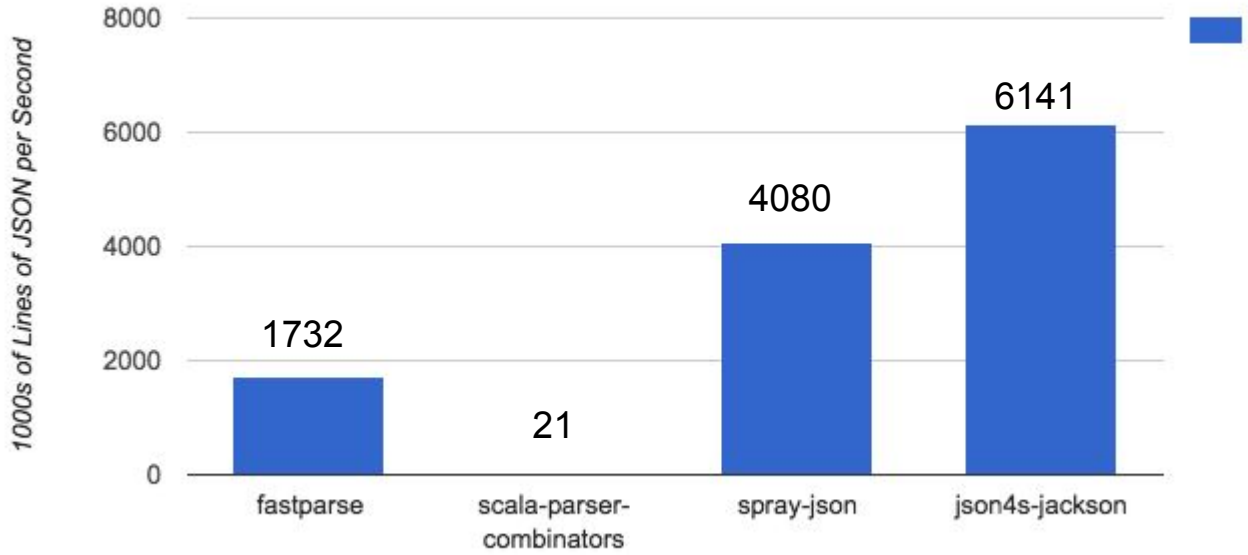
Runs on both Javascript and JVM!

Usage & Error Reporting

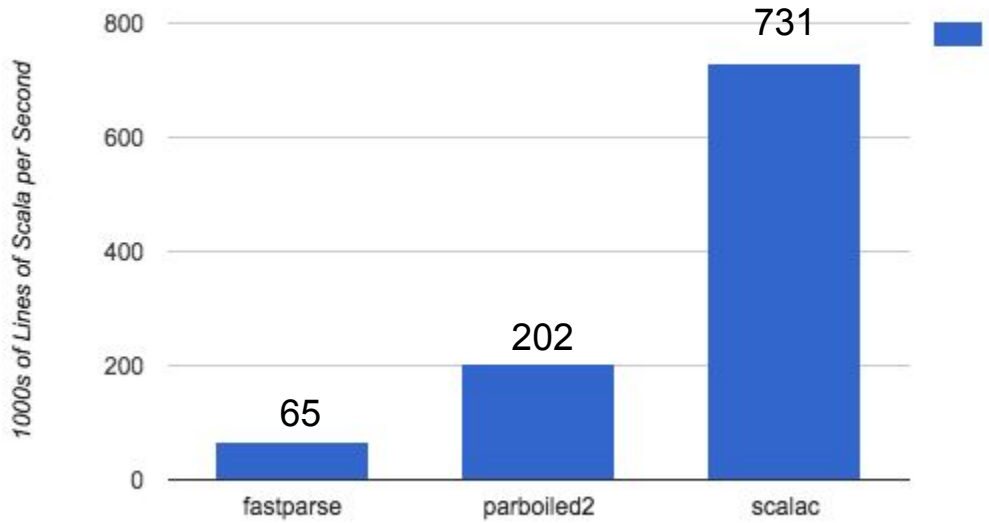
```
import fastparse.all._
val num = P( CharIn('0' to '9').rep(1) ).!.map(_.toInt)
val side = P( "(" ~! expr ~ ")" | num )
val expr: P[Int] = P( side ~ "+" ~ side ).map{case (l, r) => l + r}

> expr.parse("(1+(2+3))+4")
Success(10, index = 11)
> expr.parse("(1+(2+3x))+4")
Failure(")":7 ..."x))+4")
> expr.parse("(1+(2+3x))+4").asInstanceOf[Result.Failure].traced.trace
expr:0 / side:0 / expr:1 / side:3 / (")" | CharIn("0123456789")):7 ..."x))+4"
```

Performance

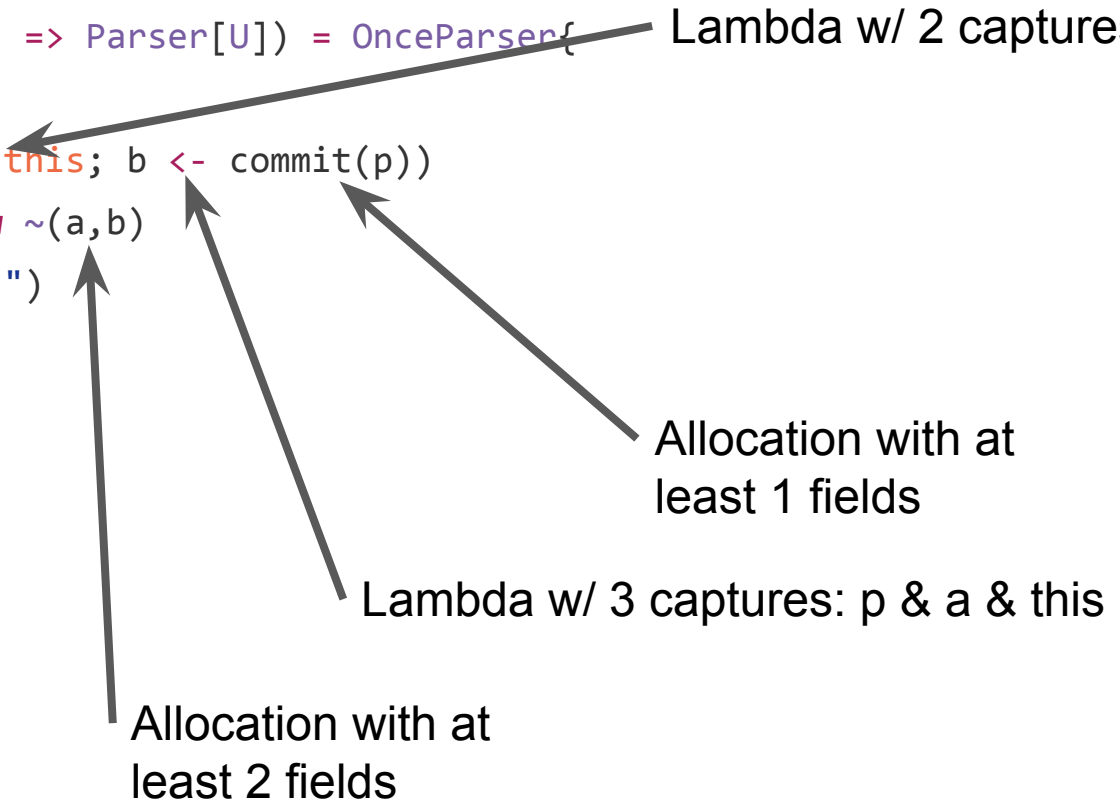


Performance



Scala-Parser-Combinator Internals

```
def ~![U](p: => Parser[U]) = OnceParser{
 (
   for(a <- this; b <- commit(p))
   yield new ~(a,b)
 ).named("~!")
}
```



FastParse Internals

```
def parseRec(cfg: ParseCtx, index: Int) = p1.parseRec(cfg, index) match{
  case f: Mutable.Failure => failMore(f, index, cfg.logDepth, traceParsers = if(cfg.traceIndex ==
-1) Nil else List(p1), cut = f.cut)
  case Mutable.Success(value0, index0, traceParsers0, cut0)  =>
    p2.parseRec(cfg, index0) match{
      case f: Mutable.Failure => failMore(
        f, index, cfg.logDepth,
        traceParsers = traceParsers0 ::: f.traceParsers,
        cut = cut | f.cut | cut0
      )
      case Mutable.Success(value1, index1, traceParsers1, cut1)  =>
       success(cfg.success, ev(value0, value1), index1, traceParsers1 ::: traceParsers0, cut1 |
cut0 | cut)
    }
}
```

All in one method

Zero allocations

Implementation Details

Straightforward recursive-descent PEG

- No fancy parsing algorithms, disambiguation, async/push-parsing, ...
- No fancy macro-optimizations or parser-transformations; WYWIWYG

Object Oriented Design

- Build your own components! Just implement `Parser[+T]`

Externally immutable, but...

- Built-in `Parser[+T]`s are optimized & fast: while-loops, bitsets, etc.
- Internally uses `Mutable.{Success[T], Failure}` to save allocations

Uses of FastPare

Examples: Math, Whitespace-handling, indentation-blocks, JSON

- http://lihaoyi.github.io/fastparse/#ExampleParsers

PythonParse: parsing a full python AST from source, including indentation-blocks

- https://github.com/lihaoyi/fastparse/tree/master/pythonparse

ScalaParse: parses Scala without generating an AST, heavily used in Ammonite

- https://github.com/lihaoyi/fastparse/tree/master/scalaparse

Scalatex: Programmable documents; uses ScalaParse & adds indentation-blocks

- https://github.com/lihaoyi/Scalatex

FastParse is...

A new Parser Combinator library for Scala

Very convenient (in code, no special build step)

Great error reporting

Bog-standard recursive-descent/PEG

“bat-out-of-hell fast” - Mark Waks

Super flexible

Runs on both Javascript and JVM!

Questions?

Code & Issues: https://github.com/lihaoyi/fastparse

Docs: https://lihaoyi.github.io/fastparse

Chat Room: https://gitter.im/lihaoyi/fastparse

Ask me about

- Hack-free indentation-parsing, semicolon-inference
- Higher-order parsers
- Monadic Parser Combinators