Hubris

subjecting Haskell to Ruby's iron will

Mark Wotton <mwotton@shimweasel.com>

September 8, 2009

I ♥ Ruby

- concise and flexible
- Big web community, many libraries
- Fun

I ♥ Haskell

- Fast (optimised native code, multicore, etc)

I ♥ Haskell

- Fast (optimised native code, multicore, etc)
- Expressive - type systems don't have to suck.

I ♥ Haskell

- Fast (optimised native code, multicore, etc)
- Expressive - type systems don't have to suck.
- Provably safe at compile time

the slide that's going to get me lynched

JRuby vs GHC

Program	Time	Memory	Source Size
reverse-complement	5	1	1/4
regex-dna	7	3	1/5
binary-trees	8	7	1
k-nucleotide	10	1	1/7
pidigits	18	18	2
n-body	26	53	1
chameneos-redux	30	24	1
fasta	31	142	1
fannkuch	45	22	1/4
spectral-norm	227	56	1/3
mandelbrot	319	3	1/2

319 times? Really?

- Haskell code written by expert hackers

319 times? Really?

- Haskell code written by expert hackers
- Longer in some cases - gone hard on optimisations even when they blow out source code size. But...

319 times? Really?

- Haskell code written by expert hackers
- Longer in some cases - gone hard on optimisations even when they blow out source code size. But...
- 319 times faster, 1/3 the memory.

more lies to outrage Myles

there are approximately twelve programmers in the world who know Haskell

Nine are working on four different compilers with extensions for solving logic puzzles in the type system (not kidding, google "haskell instant insanity")

The other three are working on six different web frameworks

Peanut butter, meet chocolate

Ruby has a heap of web frameworks, convenience libraries, well-tested integration with javascript + CSS

Haskell is smoking fast with rock solid type safety but a tiny community

Hubris is my bridge between the two

lazy, statically typed, and pure

```
clMax lim = maximumBy (comparing snd) (assocs arr)
  where arr = listArray (1,lim)
                        (0:(map depth [2..]))
        step x = if even x
                 then div x 2
                 else 3 * x + 1
        depth x = 1 + if n <= lim
                      then arr ! n
                      else depth n
          where n = step x
```

actually using it

```
require Hubris                  # my favourite line
c = Collatz.new                 # any ruby object
c.inline(haskell_string) # from above
puts c.clMax(1000000)
>> 837799
```

Making it less sucky

Needs to use jhc, because ghc can't produce dynamic libs right now (works with GHC HEAD, will be compatible with 6.12)

one-way bridge, no callbacks to Ruby

a smarter mapping layer

caching of Haskell binaries

cleanup of ruby interface

shore up support for arrays, hashes, BigInts

autoconf support to find ruby libs and includes

... lots to do

Try it out!

install GHC and JHC
git clone git://github.com/mwotton/Hubris.git
follow the README
tell me what's missing
patches very much welcome (thanks to Josh Price, James Britt and Tatsuhiro Ujihisa)

Learning Haskell

Learn You A Haskell: why the lucky stiff's academic cousin
Real World Haskell: awesome, practical introduction
haskell channel on freenode
beginners@haskell.org