Why JRuby, Why?!

Brett Giles

YYCRuby Meetup

2014-11

Ruby on Java!

- It *is* just ruby. Write it. Run it.
- .... but... you *can* call Java.
- .... and... you *can't* call C.

Why call Java?

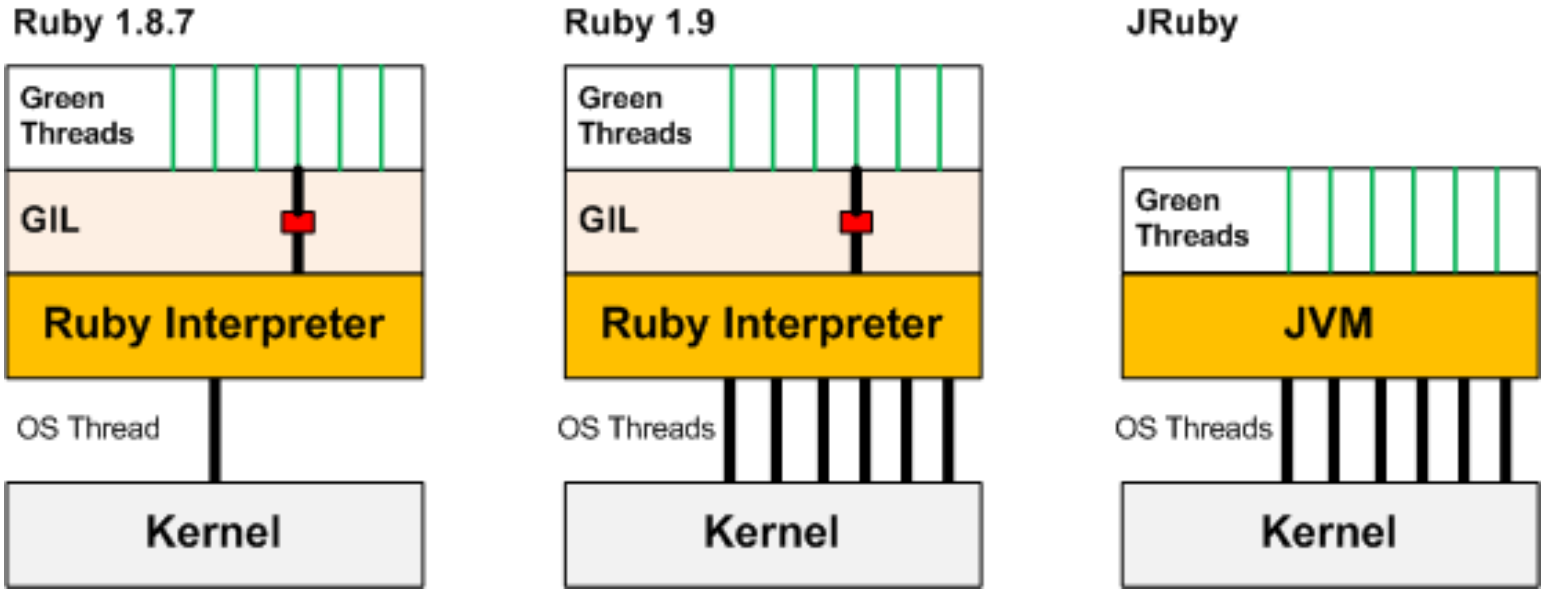
- Threads!
- WARs!
- Big conservative companies!
- Swing!

Why *not* call Java?

- Context switching (yours - not the computer's).
- JVM startup is a pain (unTDD-like). (Spork, NG-Server are answers).
- Tends to lag behind. Current production release is 1.9 compatible. 2.2 coming soonish...
- You really really *really* need the C interface. (Try harder not to :)

Threads

The dreaded Global Interpreter Lock!



But... Rubinius does not have a GIL either....

Diagram from https://www.igvita.com/2008/11/13/concurrency-is-a-myth-in-ruby/

Thread code - Ruby

```ruby
num_iterations.times do |iter|
  threads = []
  t_0 = Time.now
  (1..num_threads).each do |i|
    threads << Thread.new(i) do
      count = 0
      1_000_000.times { count += 1 }
    end
  end
  threads.each(&:join)
end
```

Thread code - JRuby - Pg 1

```
class CountMillion
  include Callable
  def call
    count = 0
    1_000_000.times { count += 1 }
  end
end
executor = ThreadPoolExecutor.new(
   num_threads, # core_pool_treads
   num_threads, # max_pool_threads
   60, # keep_alive_time
   TimeUnit::SECONDS,
   LinkedBlockingQueue.new)
```

Thread code - JRuby - Pg 2

```
num_iterations.times do |i|
  tasks = []

  t_0 = Time.now
  num_threads.times do
    task = FutureTask.new(CountMillion.new)
    executor.execute(task)
    tasks << task
  end

  # Wait for all threads to complete
  tasks.each(&:get)
end
```

DEMO

WARs and BCCs

...or “How to use Rails in a 6000+ person O&G company”.

- JRuby on Rails works. Started using it at 2.x, did some 3, other have done 4.
- A WAR is a Web ARchive and is how Java Web apps are deployed on a server, e.g. Tomcat.
- `github.com/jruby/warbler` will package your app into a warfile.
- Can use Trinidad as lightweight option — one java process, many threads.

When you are in a large conservative company, all of the above is a GOOD THING!

JRuby on Rails

Need to do a few things....

- Use `therubyrhino` instead of `therubyracer`
- Use `activerecord-jdbc-adapter` plus db of choice.
- Check the jruby wiki for alternatives if using gems with C-Extensions.
- Can use on Heroku!

Of course, must have the capability to get deploy to happen by just supplying a WAR. (Bigger companies may insist on building by themselves.)

Niceties of JRoR

We started with JRoR as it was the only choice — The nice things about this:

- Fast. Speedy. Zoom! (After startup)
- Threads - Had to be a bit careful, but Rails 4 is thread-safe.
- Memory usage is good, scales well.

Quotes and resources

On a web application that took advantage of the shared memory capabilities to provide responsive processing:

> *"Teams have tried and failed to write an application that can handle the demands of [Cooperative Speculative planning], this [JRuby] application exceeds all of our expectations"*

Resources:

- PragProg: "Using JRuby" and "Deploying with JRuby".
- The wiki and the JRuby website.
- Twitter: @headius, @tom_enebo, @jruby.

and finally, swing, swing, swing

Why? (no, really, why?)

- I had to convert a Haskell GTK based GUI to something people could build on their own.
- It had to be cross platform.
- I liked Ruby.

Options for swing:

- No framework - just use the swing classes and methods.
- Rubeus - Adds block to the swing class constructors.
- MonkeyBars - Encourages MVC design, has generators!

actually using it...

- I use rake, rspec, cucumber, simplecov, flog, flay, rubocop and guard as part of the dev process.
- I spent WAAAY too much time getting cucumber to work with swing.
- `jruby --ng-server` can be your friend.

DEMO