

# Beyond Impossible!

## How JRuby Evolved the Java Platform



Except where otherwise noted, the content of this presentation is licensed under the Creative Commons Attribution-Share Alike 3.0 United States License (<http://creativecommons.org/licenses/by-sa/3.0/us/>).

# Agenda

- Who am I?
- What is JRuby?
- What made JRuby a challenge?
  - String representation
  - Missing POSIX features
  - Loading native libraries
  - Optimizing dynamic features
  - And so on...

# Who am I

- Charles Oliver Nutter
  - `charles.nutter@sun.com`
- JRuby co-lead at Sun Microsystems since 2006
  - JRuby dev since 2004
- JVM and Java Platform advocate
  - Make JVM succeed as a language platform
  - Make languages succeed as JVM enablers
- Late-night hacker
  - Too many projects!

# What Is JRuby?

- Ruby for the JVM
  - Mostly written in Java
  - Interpreter and bytecode compiler
  - “It's Just Ruby”
  - “It's Just a JVM Language”
- A great challenge for the JVM
  - Off-platform ecosystem, community, expectations
  - Many “impossible” tasks

# That's Impossible!

Implement an off-platform set of APIs on the JVM with support for binary/character strings, POSIX access, native libraries, fast regular expressions, and libc IO semantics, while simultaneously making it run all Ruby code with native threads and performance better than the original C-based implementation...

...and do it without a language or API spec.

# Why Do It?

- Ruby is a beautiful, expressive, **fun** language
  - Heck, I just wanted to use Ruby on JVM
- Ruby community are doing things their way
  - RubyGems, Rake, Rails, Rspec, and more
- Rails changed the face of web development
  - All new frameworks are judged against Rails
- We like a challenge
  - Hard problems are more fun to solve :)

# The Plan

- No spec...so the applications are the spec!
  - Though we did pull in several partial test suites
- Started in January 2006
- Step 1: Get IRB working
- Step 2: Get RubyGems working
- Step 3: Get Rails working
- Step 4: Make it a production-quality platform :)

# IRB

- “Interactive Ruby” console
- Primary challenge: Ruby compatibility
  - eval and binding behavior
  - Variable and constant scoping
  - Simple “run and fix” process
- Secondary challenge: “readline” support
  - “nice to have” feature for line editing, etc
  - Multiple JVM libraries to choose from
  - Wrap it, bind it, and we're there

# RubyGems

- Packaging, distribution system for Ruby
- Primary challenge: YAML support
  - No existing complete YAML library
  - Ruby's library written in C
- Secondary challenge: zlib
  - Gems are tar/gzip packages
- Tertiary challenge: sockets
  - Gems are remotely installable

# YAML

- “Yet another” or “YAML ain't” markup language
- One partial YAML library for JVM
- One badly outdated YAML library in Ruby
- Ola Bini to the rescue
  - Implemented RbYaml based on Python lib
  - Implemented JvYaml based on RbYaml
- JRuby is more YAML compliant than Ruby
  - ...but not all Ruby YAML APIs supported yet

# zlib

- Luckily, `java.util.zip` covered 90%
  - Mostly just wrapped with Ruby API
  - Some fiddly bits required work
- Demonstrated some power of Java platform
  - `java.util.zip` is mostly a thin zlib wrapper
  - Simplest “native” library for us to support

# Sockets

- Java's socket APIs got things basically working
- We'll come back to IO challenges

# Rails

- Ruby web framework that changed the world
- Make-or-break for any Ruby impl
  - If you run Rails, you're for real
  - If you don't run Rails, you're not a Ruby impl
- Primary challenge: ActiveRecord (Rails' ORM)
  - Some existing pure-Ruby (slow) drivers
  - JDBC is very low-level

# ActiveRecord-JDBC

- ActiveRecord driver backend atop JDBC
  - AR separates most driver logic
  - ...but there's a lot of quirky code
- Pure-ruby driver was too complex
  - JRuby wasn't up to it yet
- Wrapped JDBC with driver API
  - Got enough working for JavaOne 2006 demo
  - ...which got Tom and I jobs at Sun

# Demo

IRB, RubyGems, Rails

# It Can Be Done!

- With Rails working, we had our proof
  - Users, community, Sun were excited
  - But it was really, really slow
  - ...and really buggy
- Time for a systematic approach
  - Performance bottlenecks
  - Differing or missing APIs
  - Unforeseen challenges?

# Strings

- JRuby started with StringBuffer
  - Obvious initial choice
  - ...but semantic differences
  - ...and Ruby strings used for **binary data** too
- Impossible task: create our own String
  - byte[]-based, copy-on-write
  - Only way to perfectly match Ruby
  - Introduced many other impossible tasks :)

# Regular Expressions

- `java.util.regex` failed against large data
- JRegex took over, but was still `char[]`-based
- Impossible task: implement our own regex
  - Oniguruma library from C world
  - 65kloc of code (including charset tables)
  - Nontrivial port, lots of C-isms
  - Marcin Mielzynski ported in about a month
    - EPIC EFFORT!
  - Best regex library on JVM

# Demonstration

Regular Expressions

# POSIX

- chmod, chown, symlink, link, fstat, and more
  - Most still aren't available
- Impossible task: add POSIX support
  - Java Native Extensions (JNA) library
  - Programmatic loading+calling of C libs
  - JNA-POSIX project
  - Multi-platform support
  - Now moving to JFFI

# Demonstration

POSIX APIs

# Native Libraries

- Equivalent Java library?
- Pure-Ruby library?
- Impossible task: load and call native libraries
  - Foreign Function Interface (FFI) library
  - Based on JNA/JFFI
  - Ruby DSL/API for binding C libraries
  - Cross-impl support
  - First of its kind on JVM

# Demonstration

FFI

# IO

- Simple java.io/java.nio wrappers at first
- Semantic differences more and more apparent
- Impossible task: implement libc semantics
  - libc-lookalike descriptor/FILE\* subsystem
  - Blocking and nonblocking IO
  - Binary and character data
  - Single API for files, sockets, pipes, subprocesses
  - I take credit for this one!

# Performance

- 10x slower wouldn't cut it
- String, Regexp, IO work helped
- Reflection-based dispatch not feasible
- Impossible task: make Ruby fast
  - Method handle generation
  - Multiple call paths for 0-N arguments
  - Bytecode compiler
  - New JVM work in progress (for Java 7?)

# Demonstration

JRuby performance

# Java Types

- Scripting APIs are cumbersome
- Many Java APIs need annotations, etc
- Impossible Task: Ruby to Java class compiler
  - Ruby has no type declarations
  - Classes are built at runtime
  - New compiler inspects runtime classes
  - Normal Java types, plus annotations!

# Demonstration

Generating Java classes from Ruby code

# Was It Worth It?

- JRuby now an alternative to Ruby
  - Ruby compatibility is “done”
  - Performance is top-notch, at or near the top
- Many production users
  - Sun, Oracle, governments, telecom
  - Small-scale pure-Ruby shops too!
- Friends and fun
  - Traveling the world, speaking and listening
  - Software challenges are fun!

# What's The Point

- Nothing is impossible on the JVM
  - If it can be done at all, it can be done on the JVM
- JRuby has expanded the Java platform
  - A whole ecosystem and community
  - Future directions for JVM and Java
- Open Source conquers all
  - Contributors are JRuby's backbone
  - You can help!

# Links

- JRuby wiki: [wiki.jruby.org](http://wiki.jruby.org)
- JRuby home: [www.jruby.org](http://www.jruby.org)
- My blog: [blog.headius.com](http://blog.headius.com)
- My email: [charles.nutter@sun.com](mailto:charles.nutter@sun.com)
- Sun's JRuby on Rails-based project site:  
[www.kenai.com](http://www.kenai.com)
- Oracle's JRoR-based “idea” site:  
[mix.oracle.com](http://mix.oracle.com)