

Pushing the Java Platform Further

Me

Charles Oliver Nutter

- headius@headius.com
- @headius
- JRuby co-lead, fulltime since 2006
- JVM enthusiast
- <http://blog.headius.com>

JRuby

- Ruby atop the JVM
 - "Just another Ruby implementation"
 - "Just another JVM language"
- A challenge for the JVM
 - Dynamic typing/dispatch
 - Own set of core classes
 - Heavy use of OS-level features

JRuby

- Ruby atop the JVM
 - "Just another Ruby implementation"
 - "Just another JVM language"
- A challenge for the JVM
 - Dynamic typing/dispatch
 - Own set of core classes
 - Heavy use of OS-level features

Custom Core Classes

What Classes?

- Array: a mutable, growable list of objects
- Hash: an ordered associative collection
- String: a collection of bytes (w/ an encoding in 1.9)
- Symbol: a named identifier
- IO: a stream of bytes (w/ an encoding in 1.9)
- File: a stream of bytes from filesystem
- Range: a range of values

Why Custom?

- Array, Hash
 - Behavior must match Ruby exactly
 - Many operations must redispach into Ruby
- String
 - Java's String is immutable
 - Java's String is UTF-16 char[] based
- Regexp
 - byte[] based String necessitates it
 - Ruby Regexp is rich in features
- IO, File
 - Blocking IO simulated atop non-blocking
 - No good abstractions over NIO

String

String

- Used for both binary and character data
 - "It's just bytes"
 - IO operations receive, return String
- Arbitrary encoding
 - Implicit encoding in 1.8 (set globally)
 - Explicit encoding in 1.9 (String aggregates Encoding)
- Mutable
 - More efficient to mutate in-place
 - Copy-on-write to reduce object churn

ByteList

<http://github.com/jruby/bytelist>

ByteList

- A list of bytes
 - Abstraction around `byte[]` + encoding
 - Similar operations to `StringBuffer`
 - CoW-capable

ByteList Demo

Regex

Regexp

- Operates against `byte[]` in `String`
 - All other engines operate against `char[]`
 - Transcoding cost == death
- Supports arbitrary encoding
 - Not all encodings transcode losslessly
 - Mismatched but compatible encodings?
- Bytecode engine
 - Avoids `StackOverflow` bugs in `java.util.regex`
 - Optimization potential
- Advanced regex features
 - Encoding-agnostic
 - Named groups

Joni

<http://github.com/jruby/joni>

Joni

- Java port of Oniguruma
- Ruby-friendly
 - `byte[]` based
 - Arbitrary encodings
 - Bytecoded engine
- Fast
 - Avoids transcoding costs
 - Sometimes faster than `j.u.regex`

Thanks to Marcin Mielzynski (github.com/lopex)!

Joni Demo

OS-level Features

OS-level Features

- Process management
 - getpid, waitpid, kill, signal, ...
- Filesystem operations
 - symlink, ownership, permissions, ...
- User-friendly IO
 - Interruptible, inherit TTY, "select" function, ...
- Binding native libs
 - Avoid hand-written native shim (JNI)
 - Route around JVM APIs
- UNIX sockets

Java Native Runtime

<http://github.com/jnr>

Java Native Runtime

A collection of libraries to make interfacing with OS-level features easier.

- Easy binding of native libs
- NIO-like IO atop file descriptors
- UNIX sockets
- POSIX filesystem operations

Big thanks to Wayne Meissner (@wmeissner)!

JNR-FFI

(foreign function interface)

<http://github.com/jnr/jnr-ffi>

JNR-FFI

- Base of the JNR libraries
- Easy binding of native functions
- Struct mapping
- Memory/pointer management
- As fast as JNI?
 - Within 10-20% (comparing invocation overhead)
 - "a viable alternative, even for reasonably performance-sensitive libraries" - @wmeissner
 - "A hell of a lot nicer than using JNI!" - @headius

JNR-FFI Demo

JNR-POSIX

<http://github.com/jnr/jnr-posix>

JNR-POSIX

- Commonly used POSIX functions not in JDK
 - Symlinks
 - Filesystem metadata
 - Ownership, permissions
 - True "exec"
 - Process management

JNR-POSIX Demo

JNR-ENXIO

(Extended Native X-platform IO)

<http://github.com/jnr/jnr-enixio>

JNR-ENXIO

- Mimics NIO
 - Same or similar API
 - Channels, Selectors, etc
- Implemented atop JNR-FFI
 - 100% Java (other than JNR-FFI)
- Multiple backends
 - poll, kqueue currently
 - epoll, win32 needed...volunteers?
- Full set of operations for *any* IO type
 - NIO can't select on stdio, files

JNR-ENXIO Demo

JNR-UNIXSOCKET

<http://github.com/jnr/jnr-unixsocket>

JNR-UNIXSOCKET

- Built atop jnr-enxio
- UNIX sockets
 - Need I say more?

JNR-UNIXSOCKET

JNR-X86ASM

(yes, it's what you think)

<http://github.com/jnr/jnr-x86asm>

JNR-X86ASM

- API for x86/x86_64 ASM generation
- Huh?
 - Down to the metal, baybee
 - Faster bindings to native functions
 - Other weird stuff...

JNR-X86ASM Demo

Ruby FFI

<http://github.com/ffi/ffi>

Ruby FFI

- Ruby DSL for binding native libraries
- Cross-impl
 - Supported on JRuby and standard Ruby
- Fun!

Ruby FFI Demo

What Have We Learned?

- The JVM can be a great native platform
- No problem is impossible to solve ;-)
- JRuby gives you all this (and more) for free

Links and Q/A

(All on Github)

<http://github/>

- [jruby](#) - JRuby organization
 - [/jruby](#) - JRuby itself
 - [/bytelist](#) - byte[] container
 - [/jcodings](#) - byte[] encoding support
 - [/joni](#) - byte[] regex
- [/jnr](#) - Java Runtime
 - [/jnr-ffi](#) - foreign function interface
 - [/jnr-posix](#) - POSIX features
 - [/jnr-enxio](#) - native IO
 - [/jnr-unixsocket](#) - UNIX sockets
 - [/jnr-x86asm](#) - x86 assembly
- [/ffi/ffi](#) - Ruby FFI