Ruby master - Feature #6590

Dealing with bigdecimal, etc gems in JRuby

06/14/2012 01:21 PM - headius (Charles Nutter)

Status:	Assigned	
Priority:	Normal	
Assignee:	hsbt (Hiroshi SHIBATA)	
Target version:		

Description

Hello!

http://jira.codehaus.org/browse/JRUBY-6704

We have a need to make the "bigdecimal" gem work for JRuby, so that
distros (like Red Hat, mentioned in the above bug) and users can have
the same gemspec for JRuby with updated bigdecimal gem references.

I'm not sure the best way to proceed.

The bigdecimal source and gemspec all come from MRI source, and are
not versioned separately. That means we can't simply share a
repository for the JRuby bits. We could maintain a forked version in
our forked "jruby/ruby" repository, but that wouldn't be part of the
bigdecimal release cycle then.

So I'm looking to you for guidance.

The BigDecimal lib is here in JRuby:
https://github.com/jruby/jruby/tree/master/src/org/jruby/ext/bigdecimal

It might be simplest if for now there's a dummy "bigdecimal" gem
pushed for JRuby that does not contain anything, since we would have
other complications if we tried to remove BigDecimal from JRuby proper
(it is referenced elsewhere int eh code, etc).

Thoughts?

- Charlie

Related issues:

Related to Ruby master - Bug #6124: remove the "spec-only gems" in Ruby 1.9.3...	**Assigned**

History

#1 - 06/14/2012 01:21 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to mrkn (Kenta Murata)

#2 - 06/26/2012 10:06 PM - mrkn (Kenta Murata)

I'm sorry I don't understand JRuby.

#3 - 06/26/2012 10:07 PM - mrkn (Kenta Murata)

- Status changed from Assigned to Third Party's Issue

#4 - 06/27/2012 12:59 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Third Party's Issue to Open

- Assignee changed from mrkn (Kenta Murata) to nahi (Hiroshi Nakamura)

If JRuby has problems because of the new bundled gems, should we not try to find a solution?

Maybe Nahi can help?

#5 - 06/27/2012 09:13 AM - hasari (Hiro Asari)

I feel that a shim for JRuby would do the job for now. Whether or not JRuby can RubyBigDecimal.java into a gem is a separate issue; I feel that this might be a case where behavioral difference between MRI and JRuby is tolerated.

#6 - 06/29/2012 02:47 PM - vo.x (Vit Ondruch)

mrkn (Kenta Murata) wrote:

> I'm sorry I don't understand JRuby.

You don't have to understand JRuby. You just need to allow the JRuby team to upload the Java version of BigDecimal. So if they do "gem install bigdecimal" using JRuby, the proper (Java) version is installed.

#7 - 07/01/2012 04:22 PM - mrkn (Kenta Murata)

I don't know how do I allow the JRuby team to uload the Java version of BigDecimal.
What should I do?

#8 - 07/02/2012 11:53 PM - Anonymous

- File noname added

On Sun, Jul 01, 2012 at 04:22:59PM +0900, mrkn (Kenta Murata) wrote:

> Issue #6590 has been updated by mrkn (Kenta Murata).
>
> I don't know how do I allow the JRuby team to uload the Java version of BigDecimal.
> What should I do?

You can add headius like this:

$ gem owner -a headius@example.org bigdecimal

But you need his rubygems.org email address.

Charlie, would this be a sufficient solution for now? I suspect that
all stdlib gems suffer from this same problem. We should think about a
better solution. Whenever a CRuby stdlib gem with native code is released,
an equivalent JRuby gem should be released too. How can we accomplish
that?

--
Aaron Patterson
http://tenderlovemaking.com/

#9 - 07/03/2012 11:59 AM - kosaki (Motohiro KOSAKI)

On Mon, Jul 2, 2012 at 10:51 AM, Aaron Patterson
tenderlove@ruby-lang.org wrote:

> On Sun, Jul 01, 2012 at 04:22:59PM +0900, mrkn (Kenta Murata) wrote:
>
>> Issue #6590 has been updated by mrkn (Kenta Murata).
>>
>> I don't know how do I allow the JRuby team to uload the Java version of BigDecimal.
>> What should I do?
>
> You can add headius like this:
>
> $ gem owner -a headius@example.org bigdecimal
>
> But you need his rubygems.org email address.
>
> Charlie, would this be a sufficient solution for now? I suspect that
> all stdlib gems suffer from this same problem. We should think about a
> better solution. Whenever a CRuby stdlib gem with native code is released,
> an equivalent JRuby gem should be released too. How can we accomplish
> that?

I dislike this idea at all. If we take it, bigdecimal gem evetually
has c, java, C#,
haskel, pascal, etc implementations when we get more and more various
ruby interpreter.

Actually, this is not bigdecimal issue. It is gem discovery issue. MRI
support two extension types,
C and Ruby. JRuby also support two extension types, Java and Ruby. gem
loader should realized
the fact and prefer to look up Java implementation when used from
JRuby. I think.

#10 - 07/03/2012 01:35 PM - vo.x (Vit Ondruch)

kosaki (Motohiro KOSAKI) wrote:

> Actually, this is not bigdecimal issue. It is gem discovery issue. MRI
> support two extension types,
> C and Ruby. JRuby also support two extension types, Java and Ruby. gem
> loader should realized
> the fact and prefer to look up Java implementation when used from
> JRuby. I think.

Actually that is exactly what RubyGems do and what Aaron is proposing. Take nokogiri [1] for example. If you are using MRI, then the nokogiri gem is taken. If it happens you are windows user, then RubyGems downloads nokogiri-x86-mingw32 and if you are JRuby user, then the nokogiri-java version is prefered. They are all just nokogiris, so RubyGems takes the gems from one location. So the MRI gem will be uploaded by mkrn and the JRuby version by headius. It should be just somehow synchronized when new gem version is released.

[1] https://rubygems.org/gems/nokogiri

#11 - 07/03/2012 05:23 PM - kosaki (Motohiro KOSAKI)

On Tue, Jul 3, 2012 at 12:35 AM, vo.x (Vit Ondruch)
v.ondruch@tiscali.cz wrote:

> Issue #6590 has been updated by vo.x (Vit Ondruch).
>
> kosaki (Motohiro KOSAKI) wrote:
>
>> Actually, this is not bigdecimal issue. It is gem discovery issue. MRI
>> support two extension types,
>> C and Ruby. JRuby also support two extension types, Java and Ruby. gem
>> loader should realized
>> the fact and prefer to look up Java implementation when used from
>> JRuby. I think.
>
> Actually that is exactly what RubyGems do and what Aaron is proposing. Take nokogiri [1] for example. If you are using MRI, then the nokogiri gem is taken. If it happens you are windows user, then RubyGems downloads nokogiri-x86-mingw32 and if you are JRuby user, then the nokogiri-java version is prefered. They are all just nokogiris, so RubyGems takes the gems from one location. So the MRI gem will be uploaded by mkrn and the JRuby version by headius. It should be just somehow synchronized when new gem version is released.

I disagree. JRuby bigdecimal is not equal with mrkn bigdecimal. It is
a forked gem. They were
maintained different maintainer and different repository. We can't
synchronized them anytime. So, No. Please drop f*cking insane idea. It
doesn work at all.

#12 - 07/03/2012 06:02 PM - vo.x (Vit Ondruch)

kosaki (Motohiro KOSAKI) wrote:

> On Tue, Jul 3, 2012 at 12:35 AM, vo.x (Vit Ondruch)
> v.ondruch@tiscali.cz wrote:
>
>> Issue #6590 has been updated by vo.x (Vit Ondruch).
>>
>> kosaki (Motohiro KOSAKI) wrote:
>>
>>> Actually, this is not bigdecimal issue. It is gem discovery issue. MRI
>>> support two extension types,
>>> C and Ruby. JRuby also support two extension types, Java and Ruby. gem

> > > loader should realized
> > > the fact and prefer to look up Java implementation when used from
> > > JRuby. I think.
> >
> > Actually that is exactly what RubyGems do and what Aaron is proposing. Take nokogiri [1] for example. If you are using MRI, then the nokogiri gem is taken. If it happens you are windows user, then RubyGems downloads nokogiri-x86-mingw32 and if you are JRuby user, then the nokogiri-java version is prefered. They are all just nokogiris, so RubyGems takes the gems from one location. So the MRI gem will be uploaded by mkrn and the JRuby version by headius. It should be just somehow synchronized when new gem version is released.
>
> I disagree. JRuby bigdecimal is not equal with mrkn bigdecimal. It is
> a forked gem. They were
> maintained different maintainer and different repository. We can't
> synchronized them anytime. So, No. Please drop f*cking insane idea. It
> doesn work at all.

It is interesting to see you position, since such collaboration works for JSON gem for example (if I am not mistaken).

Yes, you are right, the JRuby's BigDecimal is fork, but I see no reason why it shouldn't be merged, as long as people can communicate.

#13 - 07/03/2012 06:53 PM - regularfry (Alex Young)

On 02/07/12 15:51, Aaron Patterson wrote:

> On Sun, Jul 01, 2012 at 04:22:59PM +0900, mrkn (Kenta Murata) wrote:
>
> > Issue #6590 has been updated by mrkn (Kenta Murata).
> >
> > I don't know how do I allow the JRuby team to uload the Java version of BigDecimal.
> > What should I do?
>
> You can add headius like this:
>
> $ gem owner -a headius@example.org bigdecimal
>
> But you need his rubygems.org email address.
>
> Charlie, would this be a sufficient solution for now? I suspect that
> all stdlib gems suffer from this same problem. We should think about a
> better solution. Whenever a CRuby stdlib gem with native code is released,
> an equivalent JRuby gem should be released too. How can we accomplish
> that?
>
> Surely it's only necessary to coordinate both when the API changes? If
> it's just bug-fixes, they should be independent.

--
Alex

#14 - 07/04/2012 08:12 AM - headius (Charles Nutter)

Facts:

- JRuby's bigdecimal implementation is separate from MRI's
- bigdecimal is shipped as a gem for MRI now, preinstalled by default but possible to upgrade and set as a gem dependency
- Users who want the most current bigdecimal implementation on MRI may set bigdecimal as a dependency
- Libraries may also set bigdecimal as a dependency
- Without a gem that is installable on JRuby, those users and libraries will fail to install dependencies
- BigDecimal is depended on by JRuby internals and cannot be separated from JRuby (i.e. JRuby can't run without BigDecimal library built in

All we are really looking for is a way for "bigdecimal" and other preinstalled (on MRI) gem dependencies to work on JRuby. The easiest way would be to have -java platform stub gems for the native extensions in MRI.

We do not currently see any value in attempting to maintain our native extensions alongside MRI's native extensions since they are completely different codebases.

#15 - 07/14/2012 06:36 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

#16 - 11/24/2012 09:10 AM - mame (Yusuke Endoh)

- Target version set to 2.6

#17 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#18 - 05/15/2019 12:53 PM - hsbt (Hiroshi SHIBATA)

- Assignee changed from nahi (Hiroshi Nakamura) to hsbt (Hiroshi SHIBATA)

#19 - 05/15/2019 08:33 PM - Eregon (Benoit Daloze)

hsbt (Hiroshi SHIBATA) I wonder, is there a plan for addressing this?
There are now many default gems as shown by https://stdgems.org/, a fair amount of them being C extensions and not pure-Ruby.

Basically, there will be similar situations with TruffleRuby, where users try to install/update a default gem (there were not many user-submitted issues related to this yet, though).
The difference being TruffleRuby can support C extensions, although some gems will need some work to run on TruffleRuby (e.g., currently bigdecimal doesn't gem install on TruffleRuby).

For some of these gems such as stringio and fiddle, I think it might be better for TruffleRuby to keep TruffleRuby's mostly pure-Ruby implementations instead of using a C extension.
This would be possible with e.g., some RUBY_ENGINE check in both extconf.rb and the main library file, similar to how it's done for the FFI gem:
https://github.com/ffi/ffi/blob/cb3aaca588c6645b5c8186505fb42f809811055f/ext/ffi_c/extconf.rb#L3
https://github.com/ffi/ffi/blob/master/lib/ffi.rb
However, that has the downside to be potentially not fully API-compatible if there are additions in the default gem compared to the latest MRI release.

For the rest of the gems, it would probably be best for TruffleRuby to use the C extension, to reduce maintenance efforts and make sure to be compatible with MRI's version. However, that might require significant work for C extension compatibility, so it's trade-off if there is already a pure-Ruby implementation.

To give some idea, TruffleRuby can currently run these C extension default gems:
etc, json, openssl, psych and zlib.
And these C extensions default gems are untested and probably don't work yet:
bigdecimal, date, *dbm, fcntl, fiddle, io-console, stringio, strscan.

Files

noname	500 Bytes	07/02/2012	Anonymous