Ruby trunk - Bug #7645

BigDecimal#== slow when compared to true/false

01/02/2013 12:49 AM - mathie (Graeme Mathieson)

Status:	Closed		
Priority:	Normal		
Assignee:	mrkn (Kenta Murata)		
Target version:			
ruby -v:	ruby 1.9.3p327 (2012-11-10 revision 37606) [x86_64-darwin12.2.0]	**Backport:**	

Description

I was doing a spot of profiling on a Ruby on Rails application with perftools.rb and spotted that one particular chunk of code was spending a lot (nearly 60% in some tests) of its time in BigDecimal#==. It turns out that, when writing a numeric attribute in ActiveRecord, it compares the value to both true and false, and that appears to be the source of the slowness. I've reproduced this with the following sample code:

```
require 'bigdecimal'

1_000_000.times do
  BigDecimal('3') == true
end
```

This snippet takes around 7 seconds to run on my Mac. If instead we compare with a number:

```
require 'bigdecimal'

1_000_000.times do
  BigDecimal('3') == 0
end
```

the runtime drops to ~1.2 seconds. This seems suboptimal. I'm struggling to follow through the BigDecimal source code, but the profile output indicates that BigDecimal#== is causing a NameError exception to be raised, which it's then catching and returning a valid result.

I've reported this issue to the Rails tracker here: https://github.com/rails/rails/issues/8673. While there's an easy workaround for ActiveRecord (I hope, anyway!), it does strike me that BigDecimalCmp() could short-circuit and return something sensible if the comparison value is true, false or nil?

This is my first bug report to Ruby core, so apologies if it's not quite up to scratch. If you need any more information from me, please do ask. Thank you!

Related issues:

Related to Ruby trunk - Feature #7688: Error hiding with rb_rescue() on Compa...	**Closed**

Associated revisions

Revision 9aa75d08 - 01/10/2013 06:30 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@38756 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 38756 - 01/10/2013 06:30 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

Revision 38756 - 01/10/2013 06:30 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

Revision 38756 - 01/10/2013 06:30 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

Revision 38756 - 01/10/2013 06:30 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

Revision 38756 - 01/10/2013 06:30 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

Revision 38756 - 01/10/2013 06:30 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

Revision 350c448d - 01/12/2013 08:47 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@38792 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 38792 - 01/12/2013 08:47 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

Revision 38792 - 01/12/2013 08:47 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

Revision 38792 - 01/12/2013 08:47 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

Revision 38792 - 01/12/2013 08:47 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

Revision 38792 - 01/12/2013 08:47 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

Revision 38792 - 01/12/2013 08:47 AM - mrkn (Kenta Murata)

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

History

#1 - 01/02/2013 04:45 AM - al2o3cr (Matt Jones)

I've added some notes on the ticket on the Rails tracker - short story shorter, this particular case happens (AFAIK) because rb_num_coerce_cmp ends up looking for a coerce method on TrueClass.

Further insight from somebody who actually understands how this works would be appreciated. :)

#2 - 01/02/2013 05:23 AM - Eregon (Benoit Daloze)

- File coerce.patch added

Hello,

This is a nice bug report!

So, BigDecimalCmp() calls rb_num_coerce_cmp() then do_coerce(), which tries to call #coerce on true, which generates a NoMethodError, which is rescued by rb_rescue() in do_coerce().

The coerce behavior is intended and useful for custom defined math types. But do_coerce() might be optimized by using rb_check_funcall() instead of rb_funcall()+rb_rescue(), therefore not generating the exception.

This would have the side effect of not swallowing other errors happening with the call to #coerce. I think this is desirable, but I am less sure about compatibility.

It also has a small overhead for the case #coerce is defined as it first checks with #respond_to?.

Here are my numbers.
From your code sample:
before:
== 0 2.43s
== true 7.60s
after
== 0 2.62s
== true 1.56s

Without accounting the BigDecimal creation:
Ran at 2013-01-01 21:12:17 with ruby 2.0.0dev (2013-01-02 trunk 38674) [x86_64-darwin10.8.0]
before:
== 0 1.204 µs/i ± 0.020 (1.7%) <=> 830 363 ips (iterations per second)
== true 6.780 µs/i ± 0.162 (2.4%) <=> 147 482 ips
after:
== 0 1.198 µs/i ± 0.019 (1.6%) <=> 834 794 ips
== true 212.0 ns/i ± 2.189 (1.0%) <=> 4 716 687 ips

What do other committers think?
It passes test-all.

```
diff --git a/numeric.c b/numeric.c
index 52e2c36..880bef1 100644
--- a/numeric.c
+++ b/numeric.c
@@ -211,35 +211,22 @@ num_coerce(VALUE x, VALUE y)
return rb_assoc_new(y, x);
}

-static VALUE
-coerce_body(VALUE *x)
-{
```

- return rb_funcall(x[1], id_coerce, 1, x[0]); -} - -static VALUE -coerce_rescue(VALUE *x) -{

- volatile VALUE v = rb_inspect(x[1]);

- rb_raise(rb_eTypeError, "%s can't be coerced into %s",
- rb_special_const_p(x[1])?
- RSTRING_PTR(v):
- rb_obj_classname(x[1]),
- rb_obj_classname(x[0]));
- return Qnil; /* dummy */ -} - static int do_coerce(VALUE *x, VALUE *y, int err) { VALUE ary;

- VALUE a[2];

- a[0] = *x; a[1] = *y;
- ary = rb_rescue(coerce_body, (VALUE)a, err?coerce_rescue:0, (VALUE)a);
- if (!RB_TYPE_P(ary, T_ARRAY) || RARRAY_LEN(ary) != 2) {
- ary = rb_check_funcall(*y, id_coerce, 1, x);
- if (ary == Qundef && err) {

- volatile VALUE v = rb_inspect(*y);
- rb_raise(rb_eTypeError, "%s can't be coerced into %s",
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- return FALSE; /* dummy */
- }
- if (ary == Qundef || !RB_TYPE_P(ary, T_ARRAY) || RARRAY_LEN(ary) != 2) {
  if (err) {
  rb_raise(rb_eTypeError, "coerce must return [x, y]");
  }

#3 - 01/02/2013 06:43 PM - mathie (Graeme Mathieson)

Thank you for pitching in with more explanation and a patch so quickly! Much appreciated. :)

#4 - 01/03/2013 02:48 PM - mrkn (Kenta Murata)

- Status changed from Open to Assigned

- Assignee set to mrkn (Kenta Murata)

#5 - 01/10/2013 03:30 PM - mrkn (Kenta Murata)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r38756.
Graeme, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

#6 - 01/10/2013 09:16 PM - mathie (Graeme Mathieson)

And thank you for fixing it! :) What would be the chances of the change being backported to 1.9.3, too?

#7 - 01/10/2013 09:40 PM - shugo (Shugo Maeda)

- Status changed from Closed to Open

mrkn (Kenta Murata) wrote:

> - numeric.c (do_coerce): speed optimization by using rb_check_funcall instead of rb_rescue + rb_funcall. This fix is based on the patch by Benoit Daloze. [Bug #7645] [ruby-core:51213]

This change caused an error in RubySpec.

1)
Bignum#<=> with an Object returns nil if #coerce raises an exception ERROR
RuntimeError: RuntimeError
(eval):2:in coerce'
/home/shugo/src/rubyspec/core/bignum/comparison_spec.rb:103:in<=>'
/home/shugo/src/rubyspec/core/bignum/comparison_spec.rb:103:in block (3 levels) in <top (required)>'
/home/shugo/src/rubyspec/core/bignum/comparison_spec.rb:3:in'
/home/shugo/local/lib/ruby/gems/2.0.0/gems/mspec-1.5.17/bin/mspec-run:8:in `'

The code around line 103 is as follows:

```
101     it "returns nil if #coerce raises an exception" do
102       @num.should_receive(:coerce).with(@big).and_raise(RuntimeError)
103       (@big <=> @num).should be_nil
104     end
```

Is it just an implementation detail or an intentional spec change?
If so, RubySpec should be changed. Otherwise, please fix the behavior.

#8 - 01/10/2013 09:45 PM - mrkn (Kenta Murata)

> Is it just an implementation detail or an intentional spec change?
> If so, RubySpec should be changed. Otherwise, please fix the behavior.

No, it isn't intentional change.
I'll fix this soon.

#9 - 01/12/2013 06:39 AM - Eregon (Benoit Daloze)

mrkn (Kenta Murata) wrote:

> > Is it just an implementation detail or an intentional spec change?
> > If so, RubySpec should be changed. Otherwise, please fix the behavior.
>
> No, it isn't intentional change.
> I'll fix this soon.

I would be very happy to hear your opinion on this behavior.
I raised this as a separate issue: #7688.

I think it should be a spec change and this new behavior is actually helpful (and the old behavior harmful).

#10 - 01/12/2013 05:47 PM - mrkn (Kenta Murata)

- Status changed from Open to Closed

This issue was solved with changeset r38792.
Graeme, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- numeric.c (do_coerce): fix for the exceptions which the coerce method raises. The optimization done by r38756 is preserved. [Bug #7645] [ruby-core:51213]

#11 - 01/12/2013 06:34 PM - mrkn (Kenta Murata)

Eregon (Benoit Daloze) wrote:

> I would be very happy to hear your opinion on this behavior.
> I raised this as a separate issue: #7688.
>
> I think it should be a spec change and this new behavior is actually helpful (and the old behavior harmful).

I think it is most important to release version 2.0, so I fixed this to keep compatible with 1.9.3's behavior.

#12 - 01/12/2013 08:45 PM - Eregon (Benoit Daloze)

mrkn (Kenta Murata) wrote:

> Eregon (Benoit Daloze) wrote:
>
> > I would be very happy to hear your opinion on this behavior.
> > I raised this as a separate issue: #7688.
> >
> > I think it should be a spec change and this new behavior is actually helpful (and the old behavior harmful).
>
> I think it is most important to release version 2.0, so I fixed this to keep compatible with 1.9.3's behavior.

I see, you are right, it is too late for any change like this.

Files

coerce.patch	1.27 KB	01/02/2013	Eregon (Benoit Daloze)