Backport21 - Backport #9316

BigDecimal division in Ruby 2.1

12/29/2013 01:41 AM - abernardes (Andre Bernardes)

Status:	Closed	
Priority:	Normal	
Assignee:	naruse (Yui NARUSE)	

Description

=begin
When updating an app to Ruby 2.1, and I ran into the following difference between ruby 2.0.0-p353 and ruby 2.1.0 when dividing two BigDecimals:

((*Ruby 2.0.0p353:*))
2.0.0p353 :002 > (BigDecimal.new("1472.0") / BigDecimal.new("0.99")).to_f
=> 1486.868686869

((*Ruby 2.1.0p0:*))
2.1.0p0 :006 > (BigDecimal.new("1472.0") / BigDecimal.new("0.99")).to_f
=> 1487.0
=end

Associated revisions

Revision 44588 - 01/13/2014 05:29 PM - mrkn (Kenta Murata)

- ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional digit for the quotient to be compatible with bigdecimal 1.2.1 and the former. [ruby-core:59365] [#9316] [#9305]
- test/bigdecimal/test_bigdecimal.rb: tests for the above change.
- ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.

Revision 44588 - 01/13/2014 05:29 PM - mrkn (Kenta Murata)

- ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional digit for the quotient to be compatible with bigdecimal 1.2.1 and the former. [ruby-core:59365] [#9316] [#9305]
- test/bigdecimal/test_bigdecimal.rb: tests for the above change.
- ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.

Revision 44588 - 01/13/2014 05:29 PM - mrkn (Kenta Murata)

- ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional digit for the quotient to be compatible with bigdecimal 1.2.1 and the former. [ruby-core:59365] [#9316] [#9305]
- test/bigdecimal/test_bigdecimal.rb: tests for the above change.
- ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.

Revision 44588 - 01/13/2014 05:29 PM - mrkn (Kenta Murata)

- ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional digit for the quotient to be compatible with bigdecimal 1.2.1 and the former. [ruby-core:59365] [#9316] [#9305]
- test/bigdecimal/test_bigdecimal.rb: tests for the above change.
- ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.

Revision 44588 - 01/13/2014 05:29 PM - mrkn (Kenta Murata)

- ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional digit for the quotient to be compatible with bigdecimal 1.2.1 and the former. [ruby-core:59365] [#9316] [#9305]
- test/bigdecimal/test_bigdecimal.rb: tests for the above change.
- ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.

Revision 44588 - 01/13/2014 05:29 PM - mrkn (Kenta Murata)

- ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional digit for the quotient to be compatible with bigdecimal 1.2.1 and the former. [ruby-core:59365] [#9316] [#9305]
- test/bigdecimal/test_bigdecimal.rb: tests for the above change.
- ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.

Revision 16c8a6d4 - 01/27/2014 07:56 AM - naruse (Yui NARUSE)

merge revision(s) 44588: [Backport #9316]

```
    * ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional
      digit for the quotient to be compatible with bigdecimal 1.2.1 and
      the former.  [ruby-core:59365] [#9316] [#9305]

    * test/bigdecimal/test_bigdecimal.rb: tests for the above change.

    * ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_1@44711 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 44711 - 01/27/2014 07:56 AM - naruse (Yui NARUSE)

merge revision(s) 44588: [Backport #9316]

```
* ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional
  digit for the quotient to be compatible with bigdecimal 1.2.1 and
  the former.  [ruby-core:59365] [#9316] [#9305]

* test/bigdecimal/test_bigdecimal.rb: tests for the above change.

* ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.
```

History

#1 - 12/30/2013 12:42 AM - dmarkow (Dylan Markow)

=begin

The to_f appears to be irrelevant:

2.0.0p353:

```
BigDecimal.new("1472.0") / BigDecimal.new("0.99")
=> 1486.868686869
```

2.1.0:

```
BigDecimal.new("1472.0") / BigDecimal.new("0.99")
=> 1487.0
=end
```

#2 - 12/30/2013 12:46 AM - dmarkow (Dylan Markow)

=begin
Also, it appears to only happen when the right side number is less than 1:

```
BigDecimal.new("1472.0") / BigDecimal.new("0.99")
1487.0 # Should be 1486.868686869
BigDecimal.new("1472.0") / BigDecimal.new("1.0")
1472.0 # Correct
BigDecimal.new("1472.0") / BigDecimal.new("1.01")
1457.425742574257425743 # Correct
=end
```

#3 - 12/30/2013 03:30 AM - zzak (Zachary Scott)

- Status changed from Open to Assigned

- Assignee set to mrkn (Kenta Murata)

- Target version set to 2.2.0

Thanks for the report, I'll leave this one to murata-san since he knows the spec

#4 - 12/30/2013 08:06 AM - al2o3cr (Matt Jones)

Still fuzzy on the cause, but the behavior is interesting - it appears that something has changed regarding the way that non-terminating decimals are rounded.

There's also some magnitude-dependent behavior: 147.2 / 0.099 (and related variants shifting the decimals around) don't always return the same value.

Here's a little program to compare the behavior for different offsets:

https://gist.github.com/al2o3cr/8175722

Attached to that are also the results of running the script on 2.0.0 and 2.1.0.

#5 - 12/30/2013 09:19 AM - al2o3cr (Matt Jones)

Updated - I added a dump of the Prec + MaxPrec values to the output of the script. There definitely appears to be a pattern, where inputs with the same precision values produce the same results.

#6 - 12/30/2013 04:28 PM - phasis68 (Heesob Park)

It seems that the size of exponenent affects the precison of division result, but that is a wrong assumption.
I think the mininum precison is required regardless of the size of exponent.

```
irb(main):001:0> puts BigDecimal.new('1') / BigDecimal.new('3')
0.333333333E0
irb(main):002:0> puts BigDecimal.new('0.1') / BigDecimal.new('0.3')
0.0
irb(main):003:0> puts BigDecimal.new('0.01') / BigDecimal.new('0.03')
0.0
irb(main):004:0> puts BigDecimal.new('0.001') / BigDecimal.new('0.003')
0.0
```

```
irb(main):005:0> puts BigDecimal.new('0.00000000001') / BigDecimal.new('0.00000000003')
0.333333333E0
```

Here is a simple patch to ensure the minimun precison of division result.

```
diff --git a/bigdecimal.c b/bigdecimal.c.new
index e0b7c01..615c15f 100644
--- a/bigdecimal.c
+++ b/bigdecimal.c.new
@@ -1222,6 +1222,7 @@ BigDecimal_divide(Real **c, Real **res, Real **div, VALUE self, VALUE r)
*div = b;
mx = a->Prec + vabs(a->exponent);
if (mxPrec + vabs(b->exponent)) mx = b->Prec + vabs(b->exponent);
```

- ```
  if (mx<3) mx = 3;
  mx =(mx + 1) * VpBaseFig();
  GUARD_OBJ((*c), VpCreateRbObject(mx, "#0"));
  GUARD_OBJ((*res), VpCreateRbObject((mx+1) * 2 +(VpBaseFig() + 1), "#0"));
  @@ -1323,6 +1324,7 @@ BigDecimal_DoDivmod(VALUE self, VALUE r, Real **div, Real **mod)

  mx = a->Prec + vabs(a->exponent);
  if (mxPrec + vabs(b->exponent)) mx = b->Prec + vabs(b->exponent);
  ```

- ```
  if (mx<3) mx = 3;

  mx = (mx + 1) * VpBaseFig();
  GUARD_OBJ(c, VpCreateRbObject(mx, "0"));
  GUARD_OBJ(res, VpCreateRbObject((mx+1) * 2 +(VpBaseFig() + 1), "#0"));
  ```

After appling the patch, the results are all equal.

```
irb(main):001:0> puts BigDecimal.new('1') / BigDecimal.new('3')
0.33333333333333333E0
irb(main):002:0> puts BigDecimal.new('0.1') / BigDecimal.new('0.3')
0.33333333333333333E0
irb(main):003:0> puts BigDecimal.new('0.01') / BigDecimal.new('0.03')
0.33333333333333333E0
irb(main):004:0> puts BigDecimal.new('0.001') / BigDecimal.new('0.003')
0.33333333333333333E0
irb(main):005:0> puts BigDecimal.new('0.00000000001') / BigDecimal.new('0.00000000003')
0.33333333333333333E0
```

#7 - 12/31/2013 01:47 AM - ehutzelman (Eric Hutzelman)

=begin
I'm also seeing some different rounding/conversion in 2.1.0p0 when using BigDecimal and Floats:

((*Ruby 2.0.0p353*:))

```
0.25 / BigDecimal.new("0.5")
=> 0.5
_.class
=> Float
```

((*Ruby 2.1.0p0*:))

```
0.25 / BigDecimal.new("0.5")
=> 0.0
_.class
=> BigDecimal
```

=end

#8 - 01/03/2014 05:52 AM - al2o3cr (Matt Jones)

phasis68 (Heesob Park) wrote:

> Here is a simple patch to ensure the minimun precison of division result.
>
> diff --git a/bigdecimal.c b/bigdecimal.c.new

index e0b7c01..615c15f 100644
--- a/bigdecimal.c
+++ b/bigdecimal.c.new
@@ -1222,6 +1222,7 @@ BigDecimal_divide(Real **c, Real **res, Real **div, VALUE self, VALUE r)
*div = b;
mx = a->Prec + vabs(a->exponent);
if (mxPrec + vabs(b->exponent)) mx = b->Prec + vabs(b->exponent);

- if (mx<3) mx = 3;
  mx =(mx + 1) * VpBaseFig();
  GUARD_OBJ((*c), VpCreateRbObject(mx, "#0"));
  GUARD_OBJ((*res), VpCreateRbObject((mx+1) * 2 +(VpBaseFig() + 1), "#0"));
  @@ -1323,6 +1324,7 @@ BigDecimal_DoDivmod(VALUE self, VALUE r, Real **div, Real **mod)

  mx = a->Prec + vabs(a->exponent);
  if (mxPrec + vabs(b->exponent)) mx = b->Prec + vabs(b->exponent);

- if (mx<3) mx = 3;

  mx = (mx + 1) * VpBaseFig();
  GUARD_OBJ(c, VpCreateRbObject(mx, "0"));
  GUARD_OBJ(res, VpCreateRbObject((mx+1) * 2 +(VpBaseFig() + 1), "#0"));

Not a fan of this, since it's updating code that didn't change between the two releases (as far as I can tell). I'd prefer to fix the underlying reason why identical calculations in 2.0 and 2.1 aren't behaving the same way instead of just arbitrarily forcing more precision...

#9 - 01/03/2014 10:52 AM - ariveira (Alexandre Riveira)

When upgrading to ruby 2.1 calculate a percentage in our system broke
example

v1 = BigDecimal.new("0.94")
v2 = BigDecimal.new("0.97")
puts (((v2 - v1) * 100) / v1).to_f

ruby 2.0 => 3.191489362
ruby 2.1 => 3.0

#10 - 01/08/2014 08:28 AM - kwerle (Kurt Werle)

This looks like the same thing as https://bugs.ruby-lang.org/issues/9305

#11 - 01/10/2014 03:40 AM - kwerle (Kurt Werle)

It is worth noting that replacing the ruby 2.1 implementation of BigDecimal with the 2.0 implementation works and makes this issue go away. The downside is that there are a few BigDecimal tests that fail. At a glance, they all look like new BigDecimal functionality.

#12 - 01/13/2014 05:03 PM - mrkn (Kenta Murata)

This behavior change had been introduced from the version 1.2.2, which was released on 2013-11-22.

This change can be thought as a bug fix for the former behavior which calculates too many digits. I thought so when I implemented it.
For example, BigDecimal("1472.0") is 5-digit floating-point number and BigDecimal("0.99") is 2-digit floating-point number, so the result of BigDecimal("1472.0") / BigDecimal("0.99") has three exact digits and one ambiguous digit.

The reason why I implemented this behavior is that the "/" operator cannot determine the precision of the quotient generally because it cannot obtain the 2nd parameter to get the precision.

However, since the current precision system isn't strictly implemented, I think the current behavior is too strict for practical use.
For instance, BigDecimal("0.1200") cannot represent a 4-digit floating-point number in the current implementation.

So I'll fix it soon, but I'll put it back after I make the precision system more strict.

#13 - 01/13/2014 05:37 PM - mrkn (Kenta Murata)

I've fixed this and released the new bigdecimal version 1.2.4.

Please do gem install bigdecimal and check it as the following:

```
gem 'bigdecimal', '>= 1.2.4'
require 'bigdecimal'

puts BigDecimal('1472.0') / BigDecimal('0.99')
```

#14 - 01/13/2014 05:39 PM - mrkn (Kenta Murata)

- Tracker changed from Bug to Backport

- Project changed from Ruby master to Backport21

- Category deleted (lib)

- Assignee changed from mrkn (Kenta Murata) to naruse (Yui NARUSE)

- Target version deleted (2.2.0)

I think it needs to be backported to 2.1 and 2.0.

#15 - 01/15/2014 03:21 PM - mrkn (Kenta Murata)

bigdecimal 1.2.4 failed to support Ruby 1.9.3.
Please use the version 1.2.5 for Ruby 1.9.3.

#16 - 01/27/2014 07:57 AM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

Applied in changeset r44711.

---

merge revision(s) 44588: [Backport #9316]

```
* ext/bigdecimal/bigdecimal.c (BigDecimal_divide): Add an additional
  digit for the quotient to be compatible with bigdecimal 1.2.1 and
  the former.  [ruby-core:59365] [#9316] [#9305]

* test/bigdecimal/test_bigdecimal.rb: tests for the above change.

* ext/bigdecimal/bigdecimal.gemspec: bigdecimal version 1.2.4.
```