Ruby - Feature #12484

Optimizing Rational

06/12/2016 02:00 PM - tad (Tadashi Saito)

Status:	Closed
Priority:	Normal
Assignee:	tad (Tadashi Saito)
Target version:	

Description

Abstract

I optimized built-in library Rational. It is more than 3.7 times faster now in some cases.

Background

Rational is increasing its importance year by year. Ruby 1.9.2 uses Rational as internal representation of Time. Ruby 2.1 introduced Rational literal ("r" suffix).

But it's performance is not good enough because its implementation uses Ruby-level method calling with rb_funcall(), in spite of it's implemented in C since Ruby 1.9.1.

Proposal

I tried to improve its performance with decreasing rb_funcall(). They have been replaced with more direct representation as C.

Implementation

2 patches attached. 0001 is exporting some numeric.c functions (Is this OK?) that needed to 0002. 0002 is the main.

This is an example of typical modification:

```
- return f_mul(f_to_f(self), other);
+ return DBL2NUM(RFLOAT_VALUE(nurat_to_f(self)) * RFLOAT_VALUE(other));
```

In this code, f_mul() and f_to_f() calls rb_funcall(). I replaced those with * (native multiply) and nurat_to_f() (the body of Rational#to_f).

Evaluation

Performance and testing evaluation is done with r55389 of trunk.

Performance is improved in most of methods. Attached ratios.png shows times of improvement (bigger is better).
The benchmark is done with attached benchmark.rb script. result-trunk.tsv and result-optimized.tsv are raw scores.

Notably, Rational + Bignum became 3.7 times or more faster.

These patches also pass make test, make test-all and make test-rubyspec.
https://drone.io/github.com/tadd/ruby/70

Discussion

I have some concerns about compatibilities but I believe it's not a real problem.

Current implementation uses Ruby-level method calling as written above. If some user redefined Integer method in Ruby level, it effects to Rational calculation.

```
class Integer
  alias mul *
  def *(o)
    p "I'm *"
```

```
    mul(o)
  end
end

r = Rational(1<<64)
r * r
```

Current implementation prints "I'm *" but nothing printed with my patch. This is compatibility breakage in a strict sense, but I don't think it is used as valuable behavior from library users.

Other concern is my prerequisites. I assumed that Rational have a pair of ::Integer (internally bignum or fixnum) only because I couldn't define subclass of Integer to work with Rational.
If user can pass a subclass of Integer to the constructor of Rational, it may cause some weird behavior.

If I should care about above or other things, please let me know.

Summary

Most of Rational methods are optimized and those speed are up with more direct C representation.
I believe there is no real compatibility problem with my implementation.

Note that my codes are supported by a grant of Ruby Association. I thank to Ruby Association and grant committee members.
http://ruby.or.jp/en/news/20160406.html

Associated revisions

Revision 8d7c380216809ba5bd4a3eec41d7dda61f825ffa - 11/18/2016 03:31 PM - tad (Tadashi Saito)

- NEWS: Added entry for optimized Rational. [Feature #12484] [ci skip]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@56831 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 8d7c3802 - 11/18/2016 03:31 PM - tad (Tadashi Saito)

- NEWS: Added entry for optimized Rational. [Feature #12484] [ci skip]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@56831 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 06/12/2016 02:06 PM - tad (Tadashi Saito)

I also created PR for detailed information.
https://github.com/ruby/ruby/pull/1381

#2 - 06/12/2016 02:29 PM - nobu (Nobuyoshi Nakada)

- Description updated

Tadashi Saito wrote:

> Other concern is my prerequisites. I assumed that Rational have a pair of ::Integer (internally bignum or fixnum) only because I couldn't define subclass of Integer to work with Rational.
> If user can pass a subclass of Integer to the constructor of Rational, it may cause some weird behavior.

It should be impossible, as a subclass of Integer can't instantiate.

#3 - 06/13/2016 01:02 AM - mrkn (Kenta Murata)

> It should be impossible, as a subclass of Integer can't instantiate.

Users can instantiate subclasses of Integer in extension library, I think.

#4 - 06/13/2016 07:29 AM - matz (Yukihiro Matsumoto)

Accepted. I will give you committer right.

Matz.

#5 - 06/13/2016 07:41 AM - mrkn (Kenta Murata)

@ttad I'll check your patch. Please wait a moment.

#6 - 06/13/2016 08:26 AM - tad (Tadashi Saito)

Thank you so much, Matz!
I'll wait for muraken's review.

#7 - 06/13/2016 01:01 PM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

- Assignee set to tad (Tadashi Saito)

Hi, tad.

We discussed this issue on Developer Meeting June.
We hope to merge this patch by yourself with a review of other committers like mrkn.

I'm going to send an invitation of Ruby committer tomorrow.

#8 - 06/13/2016 03:34 PM - mrkn (Kenta Murata)

- File diff added

I've checked the attached patch, and I found a case that should be fixed.
It's related to Integer subclasses.

Nobuyoshi Nakada wrote:

> Tadashi Saito wrote:
>
> > Other concern is my prerequisites. I assumed that Rational have a pair of ::Integer (internally bignum or fixnum) only because I couldn't define subclass of Integer to work with Rational.
> > If user can pass a subclass of Integer to the constructor of Rational, it may cause some weird behavior.
>
> It should be impossible, as a subclass of Integer can't instantiate.

Using Bug::Integer::MyInteger class, Kernel.Rational method couldn't work correctly.
Please apply my attachment diff, and check the result of TestRational#test_new.

#9 - 06/13/2016 03:35 PM - mrkn (Kenta Murata)

- File deleted (diff)

#10 - 06/13/2016 03:36 PM - mrkn (Kenta Murata)

- File test_rational.rb.patch added

I've uploaded a wrong file.
The correct one is this.

#11 - 07/02/2016 02:24 PM - tad (Tadashi Saito)

Muraken:

Thank you for your investigation. Certainly it was reproduced,
but did you test with trunk? I got almost same error message
with and without my patch.

trunk (without my patch):

```
Rational_Test#test_new:
SystemStackError: stack level too deep
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `-@'
(snip)
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `-@'
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `convert'
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `Rational'
```

```
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `test_new'
```

with my patch:

```
Rational_Test#test_new:
SystemStackError: stack level too deep
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `-@'
(snip)
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `-@'
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `Rational'
    /home/tadashi/git/ruby/test/ruby/test_rational.rb:29:in `test_new'
```

So I believe the error is not due to my patch and it's a current spec.

Allowing Integer subclass for Kernel.Rational may (or may not) be
useful, but it's an another story IMHO.

#12 - 07/06/2016 03:19 AM - mrkn (Kenta Murata)

> but it's an another story IMHO.

I agree with you.

#13 - 11/12/2016 04:43 PM - mrkn (Kenta Murata)

- Status changed from Assigned to Closed

I've merged all of your commits into trunk with some modifications.
The commits exist between r56695 and r56761.

#14 - 11/13/2016 10:05 AM - tad (Tadashi Saito)

Thank you very much, muraken!

Files

0001-export-functions.patch	5.27 KB	06/12/2016	tad (Tadashi Saito)
0002-optimize-Rational-methods.patch	25.4 KB	06/12/2016	tad (Tadashi Saito)
ratios.png	80 KB	06/12/2016	tad (Tadashi Saito)
benchmark.rb	2.52 KB	06/12/2016	tad (Tadashi Saito)
result-optimized.tsv	1.19 KB	06/12/2016	tad (Tadashi Saito)
result-trunk.tsv	1.18 KB	06/12/2016	tad (Tadashi Saito)
test_rational.rb.patch	1.07 KB	06/13/2016	mrkn (Kenta Murata)