Ruby master - Feature #17135

Improve performance of Integer#size method

08/30/2020 01:10 PM - S_H_ (Shun Hiraoka)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>ko1 (Koichi Sasada)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

Description

Integer#size seems to show improved performance when written in ruby.

benchmark:

```ruby
# prelude
n = 42

# benchmark
size = n.size

loop_count = 20000000

# result

sh@MyComputer:~/rubydev/build$ make benchmark/integer_size.yml -e COMPARE_RUBY=~/.rbenv/shims/ruby -e BENCH_RUBY=../install/bin/ruby

# Iteration per second (i/s)

<table>
<thead>
<tr>
<th></th>
<th>compare-ruby</th>
<th>built-ruby</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>65.749M</td>
<td>87.117M</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>1.33x</td>
</tr>
</tbody>
</table>

COMPARE_RUBY is ruby 2.8.0dev (2020-08-28T10:47:29Z master 7e1fddba4a) [x86_64-linux]. BENCH_RUBY is patched.

pull request:
https://github.com/ruby/ruby/pull/3476

History

#1 - 08/30/2020 01:36 PM - S_H_ (Shun Hiraoka)
- Description updated

#2 - 08/30/2020 03:38 PM - sawa (Tsuyoshi Sawada)
- Description updated
- Subject changed from Improve performance for Integer#size method to Improve performance of Integer#size method

#3 - 08/31/2020 01:15 AM - shyouhei (Shyouhei Urabe)
- Assignee set to ko1 (Koichi Sasada)
- Status changed from Open to Assigned

The patch looks good to me. HOWEVER, let me -1 this. Integer#size HAS to be as fast as what is proposed here, without any extra hustle like this. ko1 (Koichi Sasada) any idea what is preventing it from running smoothly?

#4 - 09/22/2020 07:40 AM - S_H_ (Shun Hiraoka)
I try to improve performance of Integer#size in C code with refer to TrueClass#to_s

```c
static VALUE rb_cInteger_fix_size;

static VALUE
int_size (VALUE num)
```
if (FIXNUM_P(num)) {
    return rb_cInteger_fix_size;
} else if (RB_TYPE_P(num, T_BIGNUM)) {
    return rb_big_size_m(num);
} return Qnil;
}

void Init_Numeric (void)
{
    rb_cInteger_fix_size = INT2FIX (sizeof (long));
    rb_gc_register_mark_object (rb_cInteger_fix_size);
    rb_define_method (rb_cInteger, "size", int_size, 0);
}

benchmark:

prelude: |
    n = 42
benchmark: benchmark: |
    size: |
    n.size
loop_count: 2000000

result:

sh@MyComputer:~/rubydev/build$ make benchmark/benchmark.yml -e COMPARE_RUBY=~/.rbenv/shims/ruby -e BENCH_RUBY=../install/bin/ruby
# Iteration per second (i/s)
<table>
<thead>
<tr>
<th align="left"></th>
<th align="right">compare-ruby</th>
<th align="right">built-ruby</th>
</tr>
</thead>
<tbody>
<tr>
<td align="left">size</td>
<td align="right">56.456M</td>
<td align="right">75.074M</td>
</tr>
<tr>
<td align="left"></td>
<td align="right">1.33x</td>
<td align="right"></td>
</tr>
</tbody>
</table>

As a result, seems to be expected improve performance.

COMPARE_RUBY is ruby 3.0.0dev (2020-09-20T11:39:25Z master 84c4c7bec8) [x86_64-linux]. BENCH_RUBY is patched.

#5 - 10/02/2020 12:13 AM - mtk (Kenta Murata)
INT2FIX(sizeof(long)) is a constant expression because INT2FIX is a constant inline function that only does some arithmetic operations. So, I guess you don't need to keep that value in rb_cInteger_fix_size global variable.

#6 - 10/02/2020 01:48 AM - ko1 (Koichi Sasada)
shyouhei (Shyouhei Urabe) wrote in #note-3:

The patch looks good to me. HOWEVER, let me -1 this.
Integer#size HAS to be as fast as what is proposed here, without any extra hustle like this.
ko1 (Koichi Sasada) any idea what is preventing it from running smoothly?

what is the "extra hustle"?

#7 - 10/02/2020 02:19 AM - shyouhei (Shyouhei Urabe)
ko1 (Koichi Sasada) wrote in #note-6:

shyouhei (Shyouhei Urabe) wrote in #note-3:

The patch looks good to me. HOWEVER, let me -1 this.
Integer#size HAS to be as fast as what is proposed here, without any extra hustle like this.
ko1 (Koichi Sasada) any idea what is preventing it from running smoothly?

what is the "extra hustle"?

The proposed patch is basically Primitive.cexpr!'int_size(self)' . Why it must be faster than rb_define_method(rb_cInteger, "size", int_size, 0)? They should at least perform identically.