**Description**

Time#<=> will be faster around 60%. If internal values would have Fixnum, optimized function improves performance. ([https://github.com/ruby/ruby/blob/9b69e9fafc329aaa540d5adeb55124f020abfe3c/time.c#L57-L67](https://github.com/ruby/ruby/blob/9b69e9fafc329aaa540d5adeb55124f020abfe3c/time.c#L57-L67))

**Before**

```
user  system  total    real
1.410000  0.000000 1.410000  (1.407848)
```

**After**

```
user  system  total    real
0.880000  0.000000 0.880000   (0.886662)
```

**Test code**

```ruby
require 'benchmark'
Benchmark.bmbm do |x|
  x.report do
    t1 = Time.now
    t2 = Time.now
    10000000.times do
      t1 <=> t2
    end
  end
end
```

**Patch**

The patch is in [https://github.com/ruby/ruby/pull/1546](https://github.com/ruby/ruby/pull/1546)

**Associated revisions**

**Revision 92ea637c - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)**

- Improve Time#<=> performance
  - time.c (wcmp): use internal cmp() function for comparing internal Fixnum value in Time objects. On 64-bit machine, Time object might have Fixnum object internally by default and cmp() can compare the Fixnum objects directly.
    - Time#<=> will be faster around 60% on 64-bit machine.
  - time.c (cmp): add optimized path for comparing internal Bignum value by using rb_big_cmp() API. On 32-bit machine, Time object might have Bignum object internally by default.
    - Time#<=> will be faster around 50% on 32-bit machine.
Before

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>1.410000</td>
<td>0.000000</td>
<td>1.410000</td>
<td>1.407848</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.550000</td>
<td>0.000000</td>
<td>1.550000</td>
<td>1.549145</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>0.880000</td>
<td>0.000000</td>
<td>0.880000</td>
<td>0.886662</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.050000</td>
<td>0.000000</td>
<td>1.050000</td>
<td>1.047994</td>
</tr>
</tbody>
</table>

Test code

```ruby
require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Fixnum" do
    t1 = Time.now
    t2 = Time.now
    10000000.times do
      t1 <=> t2
    end
  end
  x.report "Bignum" do
    t1 = Time.at(2 ** 64)
    t2 = Time.at(2 ** 64 + 1)
    10000000.times do
      t1 <=> t2
    end
  end
end
```

Revision 58828 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)

Improve Time#<=> performance

- time.c (wcmp): use internal cmp() function for comparing internal Fixnum value in Time objects. On 64-bit machine, Time object might have Fixnum object internally by default and cmp() can compare the Fixnum objects directly.

  Time#<=> will be faster around 60% on 64-bit machine.

- time.c (cmp): add optimized path for comparing internal Bignum value by using rb_big_cmp() API. On 32-bit machine, Time object might have Bignum object internally by default.

  Time#<=> will be faster around 50% on 32-bit machine.

Test code

```ruby
require 'benchmark'
Benchmark.bmbm do |x|
  x.report "Fixnum" do
    t1 = Time.now
    t2 = Time.now
    10000000.times do
      t1 <=> t2
    end
  end

  x.report "Bignum" do
    t1 = Time.at(2 ** 64)
    t2 = Time.at(2 ** 64 + 1)
    10000000.times do
      t1 <=> t2
    end
  end
end
```

Revision 58828 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)

Improve Time#<=> performance

- time.c (wcmp): use internal cmp() function for comparing internal Fixnum value in Time objects. On 64-bit machine, Time object might have Fixnum object internally by default and cmp() can compare the Fixnum objects directly.

  Time#<=> will be faster around 60% on 64-bit machine.

- time.c (cmp): add optimized path for comparing internal Bignum value by using rb_big_cmp() API. On 32-bit machine, Time object might have Bignum object internally by default.

  Time#<=> will be faster around 50% on 32-bit machine.

[ruby-dev:50034] [Bug #13354] [Fix GH-1546]

Before

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>1.41</td>
<td>0.00</td>
<td>1.41</td>
<td>1.40</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.55</td>
<td>0.00</td>
<td>1.55</td>
<td>1.54</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>0.88</td>
<td>0.00</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.05</td>
<td>0.00</td>
<td>1.05</td>
<td>1.05</td>
</tr>
</tbody>
</table>

Test code

```ruby
require 'benchmark'
Benchmark.bmbm do |x|
  x.report "Fixnum" do
    t1 = Time.now
    t2 = Time.now
    10000000.times do
      t1 <=> t2
    end
  end

  x.report "Bignum" do
```
t1 = Time.at(2 ** 64)
t2 = Time.at(2 ** 64 + 1)
10000000.times do
t1 <=> t2
end
end

Revision 58828 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)

Improve Time#<=> performance

- time.c (wcmp): use internal cmp() function for comparing internal Fixnum value in Time objects. On 64-bit machine, Time object might have Fixnum object internally by default and cmp() can compare the Fixnum objects directly.
  
  Time#<=> will be faster around 60% on 64-bit machine.

- time.c (cmp): add optimized path for comparing internal Bignum value by using rb_big_cmp() API. On 32-bit machine, Time object might have Bignum object internally by default.
  
  Time#<=> will be faster around 50% on 32-bit machine.

[ruby-dev:50034] [Bug #13354] [Fix GH-1546]

Before

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>1.410000</td>
<td>0.000000</td>
<td>1.410000</td>
<td>(1.407848)</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.550000</td>
<td>0.000000</td>
<td>1.550000</td>
<td>(1.549145)</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>0.880000</td>
<td>0.000000</td>
<td>0.880000</td>
<td>(0.886662)</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.050000</td>
<td>0.000000</td>
<td>1.050000</td>
<td>(1.047994)</td>
</tr>
</tbody>
</table>

Test code

```ruby
require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Fixnum" do
    t1 = Time.now
    t2 = Time.now
    10000000.times do
      t1 <=> t2
    end
  end
  x.report "Bignum" do
    t1 = Time.at(2 ** 64)
    t2 = Time.at(2 ** 64 + 1)
    10000000.times do
      t1 <=> t2
    end
  end
end
```

History

#1 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)
- Status changed from Open to Closed

Applied in changeset trunk|r58828.
Improve Time<=> performance

- time.c (wcmp): use internal cmp() function for comparing internal Fixnum value in Time objects. On 64-bit machine, Time object might have Fixnum object internally by default and cmp() can compare the Fixnum objects directly.
  
  Time<=> will be faster around 60% on 64-bit machine.

- time.c (cmp): add optimized path for comparing internal Bignum value by using rb_big_cmp() API. On 32-bit machine, Time object might have Bignum object internally by default.

  Time<=> will be faster around 50% on 32-bit machine.

[ruby-dev:50034] [Bug #13354] [Fix GH-1546]

Before

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>1.410000</td>
<td>0.000000</td>
<td>1.410000</td>
<td>( 1.407848)</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.550000</td>
<td>0.000000</td>
<td>1.550000</td>
<td>( 1.549145)</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixnum</td>
<td>0.880000</td>
<td>0.000000</td>
<td>0.880000</td>
<td>( 0.886662)</td>
</tr>
<tr>
<td>Bignum</td>
<td>1.050000</td>
<td>0.000000</td>
<td>1.050000</td>
<td>( 1.047994)</td>
</tr>
</tbody>
</table>

Test code

```ruby
require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Fixnum" do
    t1 = Time.now
    t2 = Time.now
    10000000.times do
      t1 <=> t2
    end
  end

  x.report "Bignum" do
    t1 = Time.at(2 ** 64)
    t2 = Time.at(2 ** 64 + 1)
    10000000.times do
      t1 <=> t2
    end
  end
end
```

03/14/2022 5/5