Ruby master - Bug #13357

Improve Time#+ & Time#- performance

03/23/2017 11:08 AM - watson1978 (Shizuo Fujita)

Status: Closed
Priority: Normal
Assignee:
Target version:
ruby -v:
Backport: 2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

Improve Time#+ & Time#- performance

Time#+ & Time# will be faster around 15%.
If internal values would have Fixnum, optimized function improves performance.

Before

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+</td>
<td>0.820000</td>
<td>0.000000</td>
<td>0.820000 (0.818081)</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.810000</td>
<td>0.000000</td>
<td>0.810000 (0.813835)</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+</td>
<td>0.710000</td>
<td>0.000000</td>
<td>0.710000 (0.710241)</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.710000</td>
<td>0.010000</td>
<td>0.720000 (0.714151)</td>
</tr>
</tbody>
</table>

Test code

```
require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Time#+" do
    t = Time.now
    2000000.times do
      t += 1
    end
  end

  x.report "Time#-" do
    t = Time.now
    2000000.times do
      t -= 1
    end
  end
end
```

Patch

The patch is in https://github.com/ruby/ruby/pull/1547

Associated revisions

Revision c208d15f - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)

Improve Time#+ & Time#- performance

- time.c (wadd): use internal addv() function to calculate internal value in
Time object. On 64-bit machine, Time object might have Fixnum object internally by default and addv() can calculate Fixnum objects directly.

- time.c (wsub): use internal subv() function due the same reason in above.
  Time#+ & Time#- will be faster around 15%.

[ruby-dev:50036] [Bug #13357] [Fix GH-1547]

Before

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+</td>
<td>0.820000</td>
<td>0.000000</td>
<td>0.820000</td>
<td>( 0.818081)</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.810000</td>
<td>0.000000</td>
<td>0.810000</td>
<td>( 0.813835)</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>Time#+</td>
<td>0.710000</td>
<td>0.000000</td>
<td>0.710000</td>
<td>( 0.710241)</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.710000</td>
<td>0.010000</td>
<td>0.720000</td>
<td>( 0.714151)</td>
</tr>
</tbody>
</table>

Test code

require 'benchmark'
Benchmark.bmbm do |x|
  x.report "Time#+" do
    t = Time.now
    2000000.times do
      t += 1
    end
  end
  x.report "Time#-" do
    t = Time.now
    2000000.times do
      t -= 1
    end
  end
end

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58829 b2dd03c8-39d4-4d8f-98ff-823fe69b080e
Revision 58829 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)
Improve Time#+ & Time#- performance

- time.c (wadd): use internal addv() function to calculate internal value in Time object. On 64-bit machine, Time object might have Fixnum object internally by default and addv() can calculate Fixnum objects directly.

- time.c (wsub): use internal subv() function due the same reason in above.
  Time#+ & Time#- will be faster around 15%.

[ruby-dev:50036] [Bug #13357] [Fix GH-1547]

Before

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+</td>
<td>0.820000</td>
<td>0.000000</td>
<td>0.820000</td>
<td>( 0.818081)</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.810000</td>
<td>0.000000</td>
<td>0.810000</td>
<td>( 0.813835)</td>
</tr>
</tbody>
</table>
Test code

require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Time#+" do
    t = Time.now
    2000000.times do
      t + 1
    end
  end

  x.report "Time#-" do
    t = Time.now
    2000000.times do
      t - 1
    end
  end
end

Revision 58829 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)

Improve Time#+ & Time#- performance

- time.c (wadd): use internal addv() function to calculate internal value in
  Time object. On 64-bit machine, Time object might have Fixnum object
  internally by default and addv() can calculate Fixnum objects directly.

- time.c (wsub): use internal subv() function due the same reason in above.

  Time#+ & Time#- will be faster around 15%.

[ruby-dev:50036] [Bug #13357] [Fix GH-1547]

Before

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+ 0.820000</td>
<td>0.000000</td>
<td>0.820000</td>
<td>( 0.818081)</td>
</tr>
<tr>
<td>Time#- 0.810000</td>
<td>0.000000</td>
<td>0.810000</td>
<td>( 0.813835)</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+ 0.710000</td>
<td>0.000000</td>
<td>0.710000</td>
<td>( 0.710241)</td>
</tr>
<tr>
<td>Time#- 0.710000</td>
<td>0.010000</td>
<td>0.720000</td>
<td>( 0.714151)</td>
</tr>
</tbody>
</table>

Test code

require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Time#+" do
    t = Time.now
    2000000.times do
      t + 1
    end
  end

  x.report "Time#-" do
    t = Time.now
    2000000.times do
      t - 1
    end
  end
end

03/21/2022
Revision 58829 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)

Improve Time#+ & Time#- performance

- time.c (wadd): use internal addv() function to calculate internal value in
  Time object. On 64-bit machine, Time object might have Fixnum object
  internally by default and addv() can calculate Fixnum objects directly.

- time.c (wsub): use internal subv() function due the same reason in above.

  Time#+ & Time#- will be faster around 15%.

[ruby-dev:50036] [Bug #13357] [Fix GH-1547]

Before

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+</td>
<td>0.820000</td>
<td>0.000000</td>
<td>0.820000</td>
<td>0.818081</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.810000</td>
<td>0.000000</td>
<td>0.810000</td>
<td>0.813835</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>Time#+</td>
<td>0.710000</td>
<td>0.000000</td>
<td>0.710000</td>
<td>0.710241</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.710000</td>
<td>0.010000</td>
<td>0.720000</td>
<td>0.714151</td>
</tr>
</tbody>
</table>

Test code

```ruby
require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Time#+" do
    t = Time.now
    2000000.times do
      t + 1
    end
  end
  x.report "Time#-" do
    t = Time.now
    2000000.times do
      t - 1
    end
  end
end
```

History

#1 - 05/21/2017 03:36 AM - watson1978 (Shizuo Fujita)

- Status changed from Open to Closed

Applied in changeset trunk|58829.

Improve Time#+ & Time#- performance

- time.c (wadd): use internal addv() function to calculate internal value in
  Time object. On 64-bit machine, Time object might have Fixnum object
  internally by default and addv() can calculate Fixnum objects directly.

- time.c (wsub): use internal subv() function due the same reason in above.

  Time#+ & Time#- will be faster around 15%.

[ruby-dev:50036] [Bug #13357] [Fix GH-1547]
Before

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+</td>
<td>0.820000</td>
<td>0.000000</td>
<td>0.820000 (0.818081)</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.810000</td>
<td>0.000000</td>
<td>0.810000 (0.813835)</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time#+</td>
<td>0.710000</td>
<td>0.000000</td>
<td>0.710000 (0.710241)</td>
</tr>
<tr>
<td>Time#-</td>
<td>0.710000</td>
<td>0.010000</td>
<td>0.720000 (0.714151)</td>
</tr>
</tbody>
</table>

Test code

```ruby
require 'benchmark'

Benchmark.bmbm do |x|
  x.report "Time#+" do
    t = Time.now
    2000000.times do
      t + 1
    end
  end

  x.report "Time#-" do
    t = Time.now
    2000000.times do
      t - 1
    end
  end
end
```