Ruby - Bug #14435

Time#gettime slow performance in forked process

02/02/2018 05:28 PM - bryanp (Bryan Powell)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.5.0p0 (2017-12-25 revision 61468) [x86_64-darwin17]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

Hi,

I'm seeing poor performance in Time#gettime when executing code in a forked process. Perhaps this is expected, but after a good amount of research I could not come to a reasonable conclusion.

See comparisons below:

```
require "ruby-prof"

RubyProf.start
10_000.times { Time.now.getlocal }
RubyProf::FlatPrinter.new(RubyProf.stop).print(STDOUT)

# Measure Mode: wall_time
# Thread ID: 70211449654940
# Fiber ID: 70211441152020
# Total: 0.014502
# Sort by: self_time

#  %self      total      self      wait     child     calls  name
#  43.02      0.006     0.006     0.000     0.000    10000   Time#getlocal
#  23.78      0.005     0.003     0.000     0.001    10000   <Class::Time>#now
#  19.30      0.014     0.003     0.000     0.011        1   Integer#times
#   8.73      0.001     0.001     0.000     0.000    10000   Time#initialize
#   0.17      0.015     0.000     0.000     0.014        1   [global]#[no method]
#   0.04      0.001     0.000     0.000     0.001        4  *Kernel#require
#   0.04      0.000     0.000     0.000     0.000        2   MonitorMixin#mon_exit
#   0.04      0.000     0.000     0.000     0.000        2   MonitorMixin#mon_enter
#   0.03      0.000     0.000     0.000     0.000       16   Module#method_added
#   0.03      0.000     0.000     0.000     0.000        4   IO#set_encoding
#   0.02      0.000     0.000     0.000     0.000        2   Class#inherited
#   0.02      0.000     0.000     0.000     0.000        2   <Module::Gem>#find_unresolved_default
_spec
#   0.02      0.000     0.000     0.000     0.000        6   <Class::Thread>#current
#   0.01      0.000     0.000     0.000     0.000        2   Kernel#respond_to?
#   0.01      0.000     0.000     0.000     0.000        2   <Class::Gem::Specification>#unresolve
d_deps
#   0.01      0.000     0.000     0.000     0.000        1   Module#private
#   0.01      0.000     0.000     0.000     0.000        2   MonitorMixin#mon_check_owner
#   0.01      0.000     0.000     0.000     0.000        2   Thread::Mutex#unlock
#   0.01      0.000     0.000     0.000     0.000        1   BasicObject#singleton_method_added
#   0.01      0.000     0.000     0.000     0.000        2   Thread::Mutex#lock
```

Forked process:

```
require "ruby-prof"

pid = Process.fork {
  RubyProf.start
  10_000.times { Time.now.getlocal }
  RubyProf::FlatPrinter.new(RubyProf.stop).print(STDOUT)
```

```
}

sleep 5; Process.kill("INT", pid)

# Measure Mode: wall_time
# Thread ID: 70295801310860
# Fiber ID: 70295809457020
# Total: 1.113997
# Sort by: self_time

#  %self      total      self      wait     child     calls  name
#  98.41      1.096     1.096     0.000     0.000    10000   Time#getlocal
#   0.67      1.113     0.007     0.000     1.105        1   Integer#times
#   0.67      0.009     0.007     0.000     0.002    10000   <Class::Time>#now
#   0.15      0.002     0.002     0.000     0.000    10000   Time#initialize
#   0.00      1.114     0.000     0.000     1.114        1   [global]#[no method]
#   0.00      0.000     0.000     0.000     0.000        4   IO#set_encoding
#   0.00      0.001     0.000     0.000     0.001        4  *Kernel#require
#   0.00      0.000     0.000     0.000     0.000        2   MonitorMixin#mon_exit
#   0.00      0.000     0.000     0.000     0.000        2   MonitorMixin#mon_enter
#   0.00      0.000     0.000     0.000     0.000        6   <Class::Thread>#current
#   0.00      0.000     0.000     0.000     0.000        2   <Module::Gem>#find_unresolved_default
_spec
#   0.00      0.000     0.000     0.000     0.000        2   Class#inherited
#   0.00      0.000     0.000     0.000     0.000        2   <Class::Gem::Specification>#unresolve
d_deps
#   0.00      0.000     0.000     0.000     0.000        2   Thread::Mutex#unlock
#   0.00      0.000     0.000     0.000     0.000        2   MonitorMixin#mon_check_owner
#   0.00      0.000     0.000     0.000     0.000        2   Thread::Mutex#lock
#   0.00      0.000     0.000     0.000     0.000       16   Module#method_added
#   0.00      0.000     0.000     0.000     0.000        1   BasicObject#singleton_method_added
#   0.00      0.000     0.000     0.000     0.000        1   Module#private
#   0.00      0.000     0.000     0.000     0.000        2   Kernel#respond_to?
```

1.11s in a forked process vs 0.01s in the main process.

I can improve performance somewhat (0.47s) by explicitly setting ENV["TZ"]:

```
require "ruby-prof"

pid = Process.fork {
  ENV["TZ"] = "UTC"

  RubyProf.start
  10_000.times { Time.now.getlocal }
  RubyProf::FlatPrinter.new(RubyProf.stop).print(STDOUT)
}

sleep 5; Process.kill("INT", pid)

# Measure Mode: wall_time
# Thread ID: 70347118612140
# Fiber ID: 70347131000480
# Total: 0.469237
# Sort by: self_time

#  %self      total      self      wait     child     calls  name
#  96.17      0.451     0.451     0.000     0.000    10000   Time#getlocal
#   1.74      0.010     0.008     0.000     0.002    10000   <Class::Time>#now
#   1.49      0.468     0.007     0.000     0.461        1   Integer#times
#   0.36      0.002     0.002     0.000     0.000    10000   Time#initialize
#   0.01      0.469     0.000     0.000     0.469        1   [global]#[no method]
#   0.00      0.000     0.000     0.000     0.000        4   IO#set_encoding
#   0.00      0.001     0.000     0.000     0.001        4  *Kernel#require
#   0.00      0.000     0.000     0.000     0.000        2   MonitorMixin#mon_enter
#   0.00      0.000     0.000     0.000     0.000        2   <Module::Gem>#find_unresolved_default
_spec
```

```
#    0.00       0.000      0.000      0.000      0.000         2   MonitorMixin#mon_exit
#    0.00       0.000      0.000      0.000      0.000         2   Kernel#respond_to?
#    0.00       0.000      0.000      0.000      0.000         6   <Class::Thread>#current
#    0.00       0.000      0.000      0.000      0.000         2   Thread::Mutex#lock
#    0.00       0.000      0.000      0.000      0.000         2   Class#inherited
#    0.00       0.000      0.000      0.000      0.000         2   <Class::Gem::Specification>#unresolve
d_deps
#    0.00       0.000      0.000      0.000      0.000        16   Module#method_added
#    0.00       0.000      0.000      0.000      0.000         2   MonitorMixin#mon_check_owner
#    0.00       0.000      0.000      0.000      0.000         2   Thread::Mutex#unlock
#    0.00       0.000      0.000      0.000      0.000         1   BasicObject#singleton_method_added
#    0.00       0.000      0.000      0.000      0.000         1   Module#private
```

I am running ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-darwin17] on macOS 10.13.3 (17D47).

Happy to help in any way I can.

Bryan P.

Related issues:

Related to Ruby - Bug #14920: Backport r63857 to fix performance problem on T...	**Closed**

History

#1 - 02/06/2018 09:30 PM - tenderlovemaking (Aaron Patterson)

After searching around, it looks like this is related to tzset implementation on macOS:

https://stackoverflow.com/questions/27932330/why-is-tzset-a-lot-slower-after-forking-on-mac-os-x

This seems like something that should be fixed in macOS, I think.

#2 - 02/07/2018 07:23 PM - bryanp (Bryan Powell)

Ah, interesting. I looked around for other issues related to tzset and found this:

https://stackoverflow.com/questions/41353532/why-is-time-utc-slower-in-a-forked-process-in-ruby-on-os-x-and-not-in-python/41371753

I did confirm that the slowdown only occurs in years without leap seconds... e.g. replace Time.now with Time.new(2016, 12, 01) in the above examples and the forked version is just as fast as the non-forked version. While I do agree that the issue seems to originate in macOS, would it be worth exploring what an optimization might look like?

#3 - 07/17/2018 05:55 PM - rafaelfranca (Rafael França)

Looks like r63857 fix this issue.

Naruse-san, can we backport it to 2.4 and 2.5? It is causing some massive performance problems on MacOS.

#4 - 07/19/2018 08:10 AM - naruse (Yui NARUSE)

- Related to Bug #14920: Backport r63857 to fix performance problem on Time class in MacOs systems added

#5 - 07/19/2018 08:11 AM - naruse (Yui NARUSE)

- Status changed from Open to Closed

Handle backport on #14920, close this ticket.