Ruby trunk - Feature #4560

[PATCH] lib/net/protocol.rb: avoid exceptions in rbuf_fill

04/07/2011 11:48 AM - normalperson (Eric Wong)

Status:	Assigned
Priority:	Normal
Assignee:	normalperson (Eric Wong)
Target version:	

Description

Blindly hitting IO#read_nonblock() and raising is expensive due
to two factors:

1) method cache being scanned/cleared when the IO::WaitReadable
extended class is GC-ed
2) backtrace generation

This reduces the likelyhood of an IO::WaitReadable exception,
but spurious wakeup can still occur due to bad TCP checksums.

This optimization only applies to non-OpenSSL sockets. I am
using IO#wait here instead of IO.select() since IO#wait is not
available on OpenSSL sockets.

Related issues:

Related to Ruby trunk - Feature #5138: Add nonblocking IO that does not use e...	**Closed**	**08/02/2011**

History

#1 - 04/11/2011 07:46 AM - headius (Charles Nutter)

=begin
This is an interesting one. JRuby recently changed how we generate backtraces to using the Java backtrace as the master. This means our backtraces are as expensive to generate as a full Java backtrace for the full stack (think generating a backtrace for all Ruby and C and intermediate calls in Ruby). As a result, any algorithms that generate backtraces as part of normal flow control took a big perf hit.

On JRuby master, I've made a change that does not generate backtraces for EAGAIN, to avoid the overhead of generating it for the expected case of read_nonblock having nothing available. But it's a bit of a band-aid. The overhead from even *creating* an exception can be weigh into a tight loop over read_nonblock when there's nothing available, and of course having the backtrace disabled could annoy someone if it leaked out (JRuby points them to a flag to turn the backtraces on). Not sure what's the best long-term solution.

Also, the 1.9 practice of mixing in WaitReadable is really dreadful. It's bad enough in JRuby that it has to construct a new singleton class for every raised exception, but the cache effects in 1.9 are really painful.
=end

#2 - 04/11/2011 09:23 AM - normalperson (Eric Wong)

=begin
redmine@ruby-lang.org wrote:

> On JRuby master, I've made a change that does not generate backtraces
> for EAGAIN, to avoid the overhead of generating it for the expected
> case of read_nonblock having nothing available. But it's a bit of a
> band-aid. The overhead from even *creating* an exception can be weigh
> into a tight loop over read_nonblock when there's nothing available,
> and of course having the backtrace disabled could annoy someone if it
> leaked out (JRuby points them to a flag to turn the backtraces on).
> Not sure what's the best long-term solution.

Perhaps something similar to the kgio[1] API with IO#tryread/trywrite
(that return :wait_readable/:wait_writable Symbols) can be moved into
the core Ruby IO class (and deprecate IO#read_nonblock/write_nonblock).

> Also, the 1.9 practice of mixing in WaitReadable is really dreadful.
> It's bad enough in JRuby that it has to construct a new singleton
> class for every raised exception, but the cache effects in 1.9 are
> really painful.

Yes, I've been trying to fix that in trunk, too:

[1] http://bogomips.org/kgio - *nix-only, but the API is fully RDoc-ed

--
Eric Wong
=end

#3 - 03/25/2012 03:16 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to akr (Akira Tanaka)

#4 - 10/28/2012 11:12 PM - akr (Akira Tanaka)

- Target version changed from 2.0.0 to 2.6

#5 - 12/25/2017 06:14 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#6 - 03/08/2018 09:09 AM - mame (Yusuke Endoh)

- Assignee changed from akr (Akira Tanaka) to normalperson (Eric Wong)

normalperson (Eric Wong), I think that this issue has been fixed more elegantly by introducing read_nonblock(exception: false). Am I right?

Files

0001-lib-net-protocol.rb-avoid-exceptions-in-rbuf_fill.patch	1.3 KB	04/07/2011	normalperson (Eric Wong)