Ruby master - Feature #16476
Socket.getaddrinfo cannot be interrupted by Timeout.timeout

01/03/2020 01:49 PM - kirs (Kir Shatrov)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>Glass_saga (Masaki Matsushita)</td>
</tr>
<tr>
<td>Target version:</td>
<td>3.1</td>
</tr>
</tbody>
</table>

**Description**

It seems like the blocking syscall done by Socket.getaddrinfo blocks Ruby VM in a way that Timeout.timeout has no effect.

See reproduction steps in getaddrinfo_interrupt.rb ([gist.github.com/kirs/00c02ef92e0418578135fe0a6cbd3d7d](https://gist.github.com/kirs/00c02ef92e0418578135fe0a6cbd3d7d)). This affects all modern Ruby versions, including the latest 2.7.0.

Combined with default 10s resolv timeout on many Linux systems, this can have a very noticeable effect on production Ruby apps being not resilient to slow DNS resolutions, and being unable to fail fast even with Timeout.timeout.

While [https://bugs.ruby-lang.org/issues/15553](https://bugs.ruby-lang.org/issues/15553) improves the situation for Addrinfo.getaddrinfo, Socket.getaddrinfo is still blocking the VM and Timeout has no effect.

I'd like to discuss what could be done to make that call non-blocking for threads in Ruby VM.

**UPD:** looking closer, I can see that Socket.getaddrinfo("www.ruby-lang.org", "http") and Addrinfo.getaddrinfo("www.ruby-lang.org", "http") call non-interruptible getaddrinfo, while Addrinfo.getaddrinfo("www.ruby-lang.org", "http", timeout: 10) calls getaddrinfo_a, which is interruptible:

```ruby
# interrupts as expected
Timeout.timeout(1) do
  Addrinfo.getaddrinfo("www.ruby-lang.org", "http", timeout: 10)
end
```

I'd maybe suggest that we try to *always* use getaddrinfo_a when it's available, including in Socket.getaddrinfo. What downsides that would have?

I'd be happy to work on a patch.

**Related issues:**

- Related to Ruby master - Feature #16381: Accept resolv_timeout in Net::HTTP Open
- Related to Ruby master - Feature #17134: Add resolv_timeout to TCPSocket Open

**History**

#1 - 01/03/2020 02:05 PM - kirs (Kir Shatrov)
- Description updated

#2 - 01/03/2020 09:08 PM - kirs (Kir Shatrov)
- Description updated

#3 - 01/07/2020 03:37 PM - Dan0042 (Daniel DeLorme)

+1

This has been an issue for a very long time, and it's often been handled by installing an asynchronous DNS resolver gem, but it would be nice if it "just worked". If it's really as simple as using getaddrinfo_a, that sounds great.

#4 - 01/10/2020 02:47 AM - kirs (Kir Shatrov)

Dan0042 (Daniel DeLorme) wrote:

+1

This has been an issue for a very long time, and it's often been handled by installing an asynchronous DNS resolver gem, but it would be nice if it "just worked". If it's really as simple as using getaddrinfo_a, that sounds great.

Thanks for feedback Daniel!
I've put a PR with the suggested fix: https://github.com/ruby/ruby/pull/2827

#5 - 02/11/2020 10:43 PM - kirs (Kir Shatrov)
- File deleted (getaddrinfo_interrupt.rb)

#6 - 02/11/2020 11:00 PM - kirs (Kir Shatrov)
- Description updated

#7 - 02/27/2020 06:07 AM - mame (Yusuke Endoh)
- Backport deleted (2.5: UNKNOWN, 2.6: UNKNOWN)
- ruby -v deleted (ruby 2.7.0p0 (2019-12-25 revision 647ee6091) [x86_64-linux])
- Assignee set to Glass_saga (Masaki Matsushita)
- Status changed from Open to Assigned
- Tracker changed from Bug to Feature

We discussed this issue at the dev-meeting, and it requires Glass_saga (Masaki Matsushita)'s review.

Note:
- It is uninterruptable under a platform that getaddrinfo_a is unavailable, but this problem is not only this proposal but also timeout: option of Addrinfo.getaddrinfo().
- Interruptable version can be implemented without getaddrinfo_a: Creating pthread for getaddrinfo function and pthread_cancel when interrupted. Contribution is welcome.

#8 - 06/01/2020 10:20 AM - kirs (Kir Shatrov)
mame (Yusuke Endoh) wrote in #note-7:

We discussed this issue at the dev-meeting, and it requires Glass_saga (Masaki Matsushita)'s review.

Note:
- It is uninterruptable under a platform that getaddrinfo_a is unavailable, but this problem is not only this proposal but also timeout: option of Addrinfo.getaddrinfo().
- Interruptable version can be implemented without getaddrinfo_a: Creating pthread for getaddrinfo function and pthread_cancel when interrupted. Contribution is welcome.

Thanks for feedback!
I've opened https://github.com/ruby/ruby/pull/3171 with the approach you've described. Please let me know if I miss anything.

#9 - 08/27/2020 07:56 AM - Glass_saga (Masaki Matsushita)
- Target version set to 36
- Status changed from Assigned to Closed

merged: https://github.com/ruby/ruby/commit/2038cc6cab6ceeffef3ec3a765c70ae684f829ed

#10 - 08/27/2020 04:17 PM - Glass_saga (Masaki Matsushita)
- Related to Feature #16381: Accept resolv_timeout in Net::HTTP added

#11 - 09/29/2020 03:37 AM - hsbt (Hiroshi SHIBATA)
- Target version changed from 36 to 3.0

#12 - 12/06/2020 02:08 AM - ioquatix (Samuel Williams)
I have some feedback:

- I wish we don't use strange abbreviations like "resolv_timeout" and use correct English like "resolve_timeout".
- I'm not convinced that getaddrinfo_a is a good idea, it has a user-space thread pool and the implementation doesn't seem great.
- We will introduce asynchronous DNS resolution using a scheduler hook anyway, for Ruby 3: https://github.com/bruno-/ruby/pull/1

Adding timeouts as arguments is not particularly useful either. It's not particularly easy to compose timeouts or use a single timeout for multiple operations, and it makes the underlying implementation more complex.

That being said, I'm not against this PR. I'm just not sure it's worth the effort/complexity.
Adding timeouts as arguments is not particularly useful either. It's not particularly easy to compose timeouts or use a single timeout for multiple operations, and it makes the underlying implementation more complex.

One way around this is to use a deadline instead of a timeout:

```ruby
# or something based on Process.clock_gettime
deadline = Time.now + 5
Addinfo.getaddrinfo("docs.ruby-lang.org", "http", deadline: deadline)
Addinfo.getaddrinfo("bugs.ruby-lang.org", "http", deadline: deadline)
```

I think this approach is much better than a Timeout.timeout block. For one, Timeout.timeout requires the use of Thread.raise and results in less deterministic behavior. Additionally, a deadline option can potentially use a different approach than raising an exception, similar to IO#read_nonblock.

---

Since getaddrinfo doesn't provide nonblocking version, I think we need our own implementation of getaddrinfo_a.

Adding timeouts as arguments is not particularly useful either. It's not particularly easy to compose timeouts or use a single timeout for multiple operations, and it makes the underlying implementation more complex.

In my experience timeout is important for web applications to return a response when a DNS resolution is too slow. It's a long requested series of improvements for HTTP client like read_timeout, connect_timeout, and write_timeout.

resolv_timeout is the last piece of that.

---

How common is it to need separate timeouts for all of these? I can easily imagine the resolv_timeout as being part of the connect_timeout (i.e. resolv+connect may not exceed connect_timeout). And at least in my experience I usually want a single "general timeout" for resolv+connect+write+read. For that reason I really like Jeremy's suggestion of a single deadline argument. You can even have a single deadline for multiple http requests. This is really perfect for most uses cases that I'm familiar with.

---

I'm not convinced that getaddrinfo_a is a good idea, it has a user-space thread pool and the implementation doesn't seem great.

https://github.com/ruby/ruby/commit/2038cc6cab6ceeffef3ec3a765c760ae684f829ed is reverted because of [Bug #17220].
Both Timeout.timeout and deadline is not the essential problem of this topic. The topic this ticket handles is "getaddrinfo is not interruptable nor timeout ready". This is because getaddrinfo(3) doesn't have timeout, async, nor nonblocking API. This ticket intends to provide the underlying implementation of such feature.