Ruby master - Bug #17748

Ruby 3.0 takes a long time to resolve DNS of nonexistent domains

03/25/2021 11:07 AM - xdmx (Eric Bloom)

I'm running into a subtle bug when trying to resolve a nonexistent domain.

```
$t ruby -v
ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x86_64-linux]
$ irb
require 'resolv'
=> true
t = Time.now ; Resolv::DNS.open { |dns| dns.getresources('thisisaninvaliddomain.com', Resolv::DNS::Resource::IN::A) } ; Time.now - t
=> 0.051897333
```

This works fine and it's fast, but as soon as I try to run this on 3.0:

```
$ rbenv local 3.0.0
$ ruby -v
ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x86_64-linux]
$ irb
require 'resolv'
=> true
t = Time.now ; Resolv::DNS.open { |dns| dns.getresources('thisisaninvaliddomain.com', Resolv::DNS::Resource::IN::A) } ; Time.now - t
=> 76.314624548
t = Time.now ; Resolv::DNS.open { |dns| dns.getresources('domain.com', Resolv::DNS::Resource::IN::A) } ; Time.now - t
=> 0.081165397
```

In this case the resolver takes a very long time to return a result, and it only happens with nonexistent domains. For those that exist it's running fast.

What's weird is that the only change I do is switching from 2.7 to 3.0, and then I suddenly have this problem.

The above code was run on my local machine (Arch Linux) with Ruby installed through rbenv. I've also double checked it on a completely different machine (server running Ubuntu 18.04) that is connected from a different country and I'm having the exact same problem. In that case Ruby was compiled directly from the source and installed on the server.

So this should potentially exclude any connection issue (as locally with 2.7 is fast, and it was tried in 2 different places), the way it was installed (rbenv vs source) and the distro (Arch vs Ubuntu). I asked someone running macos and ruby 3.0 and it seems that it was fast there, so maybe it's only a linux related bug, I wasn't able to get other people to try.

Resolving the domain with resolvectl is also running fast:

```
$ time resolvectl query thisisaninvaliddomain.com
thisisaninvaliddomain.com: resolve call failed: 'thisisaninvaliddomain.com' not found
```

This works fine and it's fast, but as soon as I try to run this on 3.0:

```
```

What's weird is that the only change I do is switching from 2.7 to 3.0, and then I suddenly have this problem.

The above code was run on my local machine (Arch Linux) with Ruby installed through rbenv. I've also double checked it on a completely different machine (server running Ubuntu 18.04) that is connected from a different country and I'm having the exact same problem. In that case Ruby was compiled directly from the source and installed on the server.

So this should potentially exclude any connection issue (as locally with 2.7 is fast, and it was tried in 2 different places), the way it was installed (rbenv vs source) and the distro (Arch vs Ubuntu). I asked someone running macos and ruby 3.0 and it seems that it was fast there, so maybe it's only a linux related bug, I wasn't able to get other people to try.

Resolving the domain with resolvectl is also running fast:

```
```

This works fine and it's fast, but as soon as I try to run this on 3.0:

```
```

What's weird is that the only change I do is switching from 2.7 to 3.0, and then I suddenly have this problem.

The above code was run on my local machine (Arch Linux) with Ruby installed through rbenv. I've also double checked it on a completely different machine (server running Ubuntu 18.04) that is connected from a different country and I'm having the exact same problem. In that case Ruby was compiled directly from the source and installed on the server.

So this should potentially exclude any connection issue (as locally with 2.7 is fast, and it was tried in 2 different places), the way it was installed (rbenv vs source) and the distro (Arch vs Ubuntu). I asked someone running macos and ruby 3.0 and it seems that it was fast there, so maybe it's only a linux related bug, I wasn't able to get other people to try.

Resolving the domain with resolvectl is also running fast:

```
```

This works fine and it's fast, but as soon as I try to run this on 3.0:

```
```

What's weird is that the only change I do is switching from 2.7 to 3.0, and then I suddenly have this problem.

The above code was run on my local machine (Arch Linux) with Ruby installed through rbenv. I've also double checked it on a completely different machine (server running Ubuntu 18.04) that is connected from a different country and I'm having the exact same problem. In that case Ruby was compiled directly from the source and installed on the server.

So this should potentially exclude any connection issue (as locally with 2.7 is fast, and it was tried in 2 different places), the way it was installed (rbenv vs source) and the distro (Arch vs Ubuntu). I asked someone running macos and ruby 3.0 and it seems that it was fast there, so maybe it's only a linux related bug, I wasn't able to get other people to try.

Resolving the domain with resolvectl is also running fast:

```
```

This is a follow up for commit 33/fb966197f11 ("Remove sender/message_id"
As the @senders instance variable is also used for tracking transaction ID allocation, simply removing an entry without releasing the ID would eventually deplete the ID space and cause Resolv::DNS.allocate_request_id to hang.

It seems the intention of the code was to check that the received DNS message is actually the response for the question made within the method earlier. Let's have it actually do so.

[Bug #12838] https://bugs.ruby-lang.org/issues/12838
[Bug #17748] https://bugs.ruby-lang.org/issues/17748

https://github.com/ruby/resolv/commit/53ca9c9209

Revision 9c0df2e8 - 05/22/2021 06:20 AM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 9edc162583a4f68532239f6249745ad9b518cbe: [Backport #17781]

lib/resolv.rb | 6 +++---
1 file changed, 3 insertions(+), 3 deletions(-)

Revision 87d02eac - 06/01/2021 04:40 PM - NAKAMURA Usaku
merge revision(s) 9edc162583a4f68532239f6249745ad9b518cbe: [Backport #17781]

lib/resolv.rb | 6 +++---
1 file changed, 3 insertions(+), 3 deletions(-)

Revision 83c255e8 - 07/03/2021 04:56 PM - usa (Usaku NAKAMURA)
merge revision(s) 9edc1625: [Backport #17781]

lib/resolv.rb | 6 +++---
1 file changed, 3 insertions(+), 3 deletions(-)
As the @senders instance variable is also used for tracking transaction ID allocation, simply removing an entry without releasing the ID would eventually deplete the ID space and cause Resolv::DNS.allocate_request_id to hang.

It seems the intention of the code was to check that the received DNS message is actually the response for the question made within the method earlier. Let's have it actually do so.

[Bug #12838] https://bugs.ruby-lang.org/issues/12838
[Bug #17748] https://bugs.ruby-lang.org/issues/17748

https://github.com/ruby/resolv/commit/53ca9c9209
---

lib/resolv.rb | 6 +++---
1 file changed, 3 insertions(+), 3 deletions(-)

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@67946 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 67946 - 07/03/2021 04:56 PM - usa (Usaku NAKAMURA)
merge revision(s) 9edc1625: [Backport #17781]
[ruby/resolv] Fix confusion of received response message

This is a follow up for commit 33fb966197f1 ("Remove sender/message_id pair after response received in resolv", 2020-09-11).

As the @senders instance variable is also used for tracking transaction ID allocation, simply removing an entry without releasing the ID would eventually deplete the ID space and cause Resolv::DNS.allocate_request_id to hang.

It seems the intention of the code was to check that the received DNS message is actually the response for the question made within the method earlier. Let's have it actually do so.

[Bug #12838] https://bugs.ruby-lang.org/issues/12838
[Bug #17748] https://bugs.ruby-lang.org/issues/17748

https://github.com/ruby/resolv/commit/53ca9c9209
---

lib/resolv.rb | 6 +++---
1 file changed, 3 insertions(+), 3 deletions(-)

History

#1 - 03/25/2021 12:16 PM - byroot (Jean Boussier)
I was able to reproduce on both 3.0.0-p0 as well as the current ruby_3_0 branch.

However it seems to be limited to the linux builds, it does not repo on macOS.

#2 - 03/25/2021 12:28 PM - byroot (Jean Boussier)
I ran a quick profiler:

All the time is spent waiting on the socket to be readable:

```
Mode: wall(1000)
Samples: 75017 (0.00% miss rate)
GC: 1 (0.00%)
```

```
TOTAL (pct) SAMPLES (pct) FRAME
75016 (100.0%) 75016 (100.0%) IO#wait_readable
```

The long time it takes it likely the default socket timeout of the system.

That wait_readable is in Resolv::DNS::Requester#request

```
IO#wait_readable (<cfunc>:1)
| samples: 75016 self (100.0%) / 75016 total (100.0%)
```

09/12/2021
Likely this one: https://github.com/ruby/ruby/blob/ad8842c06d26ee634f90008eefed1cd4d76342df/lib/resolv.rb#L677

#3 - 03/25/2021 12:41 PM - byroot (Jean Boussier)
The bug seem to be introduced by https://github.com/ruby/ruby/commit/9682db065158da5fa4ec8a3bc267da45b429b92c

Reverting it makes the resolution complete fast.
Ref: https://bugs.ruby-lang.org/issues/12838
cc jeremyevans0 (Jeremy Evans)

#4 - 03/25/2021 03:04 PM - jeremyevans0 (Jeremy Evans)
I couldn't reproduce the bug on Windows or OpenBSD. Since the bug appears to be limited to Linux, maybe we can just revert the behavior on Linux. #12838 is a corner case, certainly the timeout for invalid domains is worse. Here's a pull request for that approach: https://github.com/ruby/resolv/pull/7

Someone who runs Linux may want to develop a better fix that doesn't timeout and handles duplicate responses.

#5 - 03/25/2021 06:36 PM - byroot (Jean Boussier)

maybe we can just revert the behavior on Linux. #12838 is a corner case, certainly the timeout for invalid domains is worse.

I agree.

Someone who runs Linux

I was able to reproduce using Docker on macOS, I presume Docker on Windows would work as well. For whoever would like to find a better fix.

#6 - 03/26/2021 09:53 AM - rhenium (Kazuki Yamaguchi)
I was able to reproduce the issue on my Linux box and was digging into this. There actually seem to be two issues behind.

- If a search list is not given to Resolv::DNS and also the local domain name of the system is not set (Socket.gethostname is not fully qualified), the same query for the domain name is generated twice.
  https://github.com/ruby/resolv/pull/8
- The change in [Bug #12838] didn't fully resolve the root issue.
  https://github.com/ruby/resolv/pull/9

#7 - 04/08/2021 12:53 PM - nehresman (Nathan Ehresman)
Since this was backported and included in 2.7.3 it impacts people upgrading from 2.7.2 as well. We were just bitten by this when upgrading to 2.7.3 to address the unrelated CVE-2021-28965.

#8 - 05/11/2021 03:40 AM - rhenium (Kazuki Yamaguchi)
- Status changed from Open to Closed

Applied in changeset git@edc162583a4f685322239f6249745ad9b518cbe.

[ruby/resolv] Fix confusion of received response message

This is a follow up for commit 33fb96619711 ("Remove sender/message_id pair after response received in resolv", 2020-09-11).

As the @senders instance variable is also used for tracking transaction ID allocation, simply removing an entry without releasing the ID would eventually deplete the ID space and cause Resolv::DNS.allocate_request_id to hang.

It seems the intention of the code was to check that the received DNS...
message is actually the response for the question made within the method earlier. Let's have it actually do so.

[Bug #12838] https://bugs.ruby-lang.org/issues/12838
[Bug #17748] https://bugs.ruby-lang.org/issues/17748

https://github.com/ruby/resolv/commit/53ca9c9209