Ruby - Bug #15499

Breaking behavior on ruby 2.6: rb_thread_call_without_gvl doesn't invoke unblock_function when used on the main thread

01/03/2019 01:37 AM - apolcyn (alex polcyn)

Status:	Assigned		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	master	**Backport:**	2.4: DONTNEED, 2.5: DONTNEED, 2.6: DONE

Description

This issue was noticed when trying to add ruby 2.6 support to the "grpc" ruby gem (this gem is a native C-extension), and was caught by a unit test.

There are several APIs on the grpc ruby gem (https://github.com/grpc/grpc/tree/master/src/ruby) that invoke "rb_thread_call_without_gvl" on the current thread, doing a blocking operation in the "without gvl" callback and cancel that blocking operation in the "unblocking function". These APIs work in ruby versions prior to ruby 2.6 (e.g. ruby 2.5), but have problems when used on ruby 2.6

Minimal repro:

My system:

```
> lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description: Debian GNU/Linux 9.6 (stretch)
Release: 9.6
Codename: stretch

> ruby -v
ruby 2.6.0p0 (2018-12-25 revision 66547) [x86_64-linux

# I installed ruby 2.6.0 with rvm - https://rvm.io/rvm/install

> GRPC_CONFIG=dbg gem install grpc --platform ruby # build grpc gem from source with debug symbols
```

ruby script, "repro.rb" that looks like this:

```
require 'grpc'

ch = GRPC::Core::Channel.new('localhost:1234', {}, :this_channel_is_insecure)
ch.watch_connectivity_state(ch.connectivity_state, Time.now + 360)
```

Run "ruby repro.rb" with an interactive shell, and it will hang there. At this point, ctrl^C the process, and it will not terminate.
What should happen is this unblocking func should be invoked:
https://github.com/grpc/grpc/blob/master/src/ruby/ext/grpc/rb_channel.c#L354, but as seen with logging or debuggers, that unblocking func is never ran. Thus the blocking operation never completes and the main thread is stuck.

When the same repro.rb is ran on e.g. ruby 2.5.3 or ruby 2.4.1, the blocking operation is unblocked and the process terminates, as expected, when sending it a SIGINT.

Also note that if the blocking operation is put in a background thread, e.g. with this script:

```
require 'grpc'

th = Thread.new do
  ch = GRPC::Core::Channel.new('localhost:1234', {}, :this_channel_is_insecure)
  ch.watch_connectivity_state(ch.connectivity_state, Time.now + 360)
end
th.join
```

then "unblocking" functions will in fact be invoked upon sending the process a SIGINT, so this looks like a problem specifically with rb_thread_call_without_gvl being used on the main thread.

Please let me know and I can provide more details or alternative repro cases.

Thanks in advance.

Associated revisions

Revision 9e66910b3bd85de32e95cf019ed285a36aecfd9e - 01/04/2019 12:53 PM - Eric Wong

thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

We need another native thread to call some unblocking functions
which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS. Instead of a
permanent thread in <= 2.5, we can now rely on the thread cache
feature to perform interrupts.

[ruby-core:90865] [Bug #15499]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66708 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 9e66910b3bd85de32e95cf019ed285a36aecfd9e - 01/04/2019 12:53 PM - Eric Wong

thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

We need another native thread to call some unblocking functions
which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS. Instead of a
permanent thread in <= 2.5, we can now rely on the thread cache
feature to perform interrupts.

[ruby-core:90865] [Bug #15499]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66708 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 9e66910b - 01/04/2019 12:53 PM - Eric Wong

thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

We need another native thread to call some unblocking functions
which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS. Instead of a
permanent thread in <= 2.5, we can now rely on the thread cache
feature to perform interrupts.

[ruby-core:90865] [Bug #15499]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66708 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 23444302d9200bcc41ce279a529f73cad63c3f05 - 01/04/2019 01:14 PM - Eric Wong

introduce rb_nogvl C-API to mark ubf as async-signal-safe

zlib and bignum both contain unblocking functions which are
async-signal-safe and do not require spawning additional
threads.

We can execute those functions directly in signal handlers
without incurring overhead of extra threads, so provide C-API
users the ability to deal with that. Other C-API users may
have similar need.

This flexible API can supercede existing uses of
rb_thread_call_without_gvl and rb_thread_call_without_gvl2 by
introducing a flags argument to control behavior.

Note: this API is NOT finalized. It needs approval from other
committers. I prefer shorter name than previous
rb_thread_call_without_gvl* functions because my eyes requires
big fonts.

[Bug #15499]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66712 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 23444302d9200bcc41ce279a529f73cad63c3f05 - 01/04/2019 01:14 PM - Eric Wong

introduce rb_nogvl C-API to mark ubf as async-signal-safe

zlib and bignum both contain unblocking functions which are
async-signal-safe and do not require spawning additional
threads.

We can execute those functions directly in signal handlers
without incurring overhead of extra threads, so provide C-API
users the ability to deal with that. Other C-API users may
have similar need.

This flexible API can supercede existing uses of
rb_thread_call_without_gvl and rb_thread_call_without_gvl2 by
introducing a flags argument to control behavior.

Note: this API is NOT finalized. It needs approval from other
committers. I prefer shorter name than previous
rb_thread_call_without_gvl* functions because my eyes requires
big fonts.

[Bug #15499]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66712 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 23444302 - 01/04/2019 01:14 PM - Eric Wong

introduce rb_nogvl C-API to mark ubf as async-signal-safe

zlib and bignum both contain unblocking functions which are
async-signal-safe and do not require spawning additional
threads.

We can execute those functions directly in signal handlers
without incurring overhead of extra threads, so provide C-API
users the ability to deal with that. Other C-API users may
have similar need.

This flexible API can supercede existing uses of
rb_thread_call_without_gvl and rb_thread_call_without_gvl2 by
introducing a flags argument to control behavior.

Note: this API is NOT finalized. It needs approval from other
committers. I prefer shorter name than previous
rb_thread_call_without_gvl* functions because my eyes requires
big fonts.

[Bug #15499]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66712 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision fbad5b97e8a42304b73b6141b57fb8ac45565ec2 - 01/29/2019 05:31 AM - naruse (Yui NARUSE)

merge revision(s) 66708: [Backport #15499]

```
    thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

    We need another native thread to call some unblocking functions
    which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS.  Instead of a
    permanent thread in <= 2.5, we can now rely on the thread cache
    feature to perform interrupts.

    [ruby-core:90865] [Bug #15499]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@66940 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision fbad5b97e8a42304b73b6141b57fb8ac45565ec2 - 01/29/2019 05:31 AM - naruse (Yui NARUSE)

merge revision(s) 66708: [Backport #15499]

```
    thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

    We need another native thread to call some unblocking functions
    which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS.  Instead of a
    permanent thread in <= 2.5, we can now rely on the thread cache
```

```
    feature to perform interrupts.

    [ruby-core:90865] [Bug #15499]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@66940 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision fbad5b97 - 01/29/2019 05:31 AM - naruse (Yui NARUSE)

merge revision(s) 66708: [Backport #15499]

```
    thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

    We need another native thread to call some unblocking functions
    which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS.  Instead of a
    permanent thread in <= 2.5, we can now rely on the thread cache
    feature to perform interrupts.

    [ruby-core:90865] [Bug #15499]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@66940 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision ff98931daca1265e5bd33025d160c77e452c399e - 03/25/2020 04:19 PM - ko1 (Koichi Sasada)

check flags passed to rb_nogvl()

RB_NOGVL_UBF_ASYNC_SAFE is wrongly specified because flags
is not checked.
[Bug #15499] 23444302

Revision ff98931daca1265e5bd33025d160c77e452c399e - 03/25/2020 04:19 PM - ko1 (Koichi Sasada)

check flags passed to rb_nogvl()

RB_NOGVL_UBF_ASYNC_SAFE is wrongly specified because flags
is not checked.
[Bug #15499] 23444302

Revision ff98931d - 03/25/2020 04:19 PM - ko1 (Koichi Sasada)

check flags passed to rb_nogvl()

RB_NOGVL_UBF_ASYNC_SAFE is wrongly specified because flags
is not checked.
[Bug #15499] 23444302

Revision e4efca87ba1aa5b6a94b9007040ac3e783c26b43 - 03/26/2020 12:18 PM - ko1 (Koichi Sasada)

check flags passed to rb_nogvl()

RB_NOGVL_UBF_ASYNC_SAFE is wrongly specified because flags
is not checked.
[Bug #15499] 23444302

(cherry picked from commit ff98931daca1265e5bd33025d160c77e452c399e)

Revision e4efca87ba1aa5b6a94b9007040ac3e783c26b43 - 03/26/2020 12:18 PM - ko1 (Koichi Sasada)

check flags passed to rb_nogvl()

RB_NOGVL_UBF_ASYNC_SAFE is wrongly specified because flags
is not checked.
[Bug #15499] 23444302

(cherry picked from commit ff98931daca1265e5bd33025d160c77e452c399e)

Revision e4efca87 - 03/26/2020 12:18 PM - ko1 (Koichi Sasada)

check flags passed to rb_nogvl()

RB_NOGVL_UBF_ASYNC_SAFE is wrongly specified because flags
is not checked.
[Bug #15499] 23444302

(cherry picked from commit ff98931daca1265e5bd33025d160c77e452c399e)

History

#1 - 01/03/2019 02:03 AM - normalperson (Eric Wong)

> https://bugs.ruby-lang.org/issues/15499

Probably my fault with timer-thread elimination :<

I think we'll need to dynamically enable timer-thread
for certain unblock functions.

Can you try compiling with "cppflags=-DUBF_TIMER=2" ?

#2 - 01/03/2019 09:03 AM - normalperson (Eric Wong)

> Can you try compiling with "cppflags=-DUBF_TIMER=2" ?

Nevermind, both need fixing...

#3 - 01/03/2019 07:49 PM - apolcyn (alex polcyn)

Thanks for the quick look! I didn't get a chance to try out UBF_TIMER=2 before your last comment, but let me know if there's something else to try.

#4 - 01/03/2019 10:42 PM - normalperson (Eric Wong)

apolcyn@google.com wrote:

> Thanks for the quick look! I didn't get a chance to try out
> UBF_TIMER=2 before your last comment, but let me know if
> there's something else to try.

Sorry for the breakage. For now, you can workaround this
breakage by spawning a do-nothing thread to handle signals:

Thread.new { sleep }

I'm slowly working on a permanent fix which won't increase
overhead for the majority of use cases.

#5 - 01/04/2019 12:53 PM - normalperson (Eric Wong)

- Status changed from Open to Closed

Applied in changeset trunk|r66708.

---

thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

We need another native thread to call some unblocking functions
which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS. Instead of a
permanent thread in <= 2.5, we can now rely on the thread cache
feature to perform interrupts.

[ruby-core:90865] [Bug #15499]

#6 - 01/04/2019 01:22 PM - normalperson (Eric Wong)

- Status changed from Closed to Assigned

- Assignee set to ko1 (Koichi Sasada)

ko1: can you check the rb_nogvl C-API introduced in r66712?
Thanks.

#7 - 01/04/2019 01:22 PM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

apolcyn@google.com wrote:

> Thanks for the quick look! I didn't get a chance to try out UBF_TIMER=2 before your last comment, but let me know if there's something else to try.

Sorry for the breakage. For now, you can workaround this breakage by spawning a do-nothing thread to handle signals:

r66708 should fix the breakage in your particular case.
Again, deeply sorry for the breakage.

> Thread.new { sleep }

I'm slowly working on a permanent fix which won't increase overhead for the majority of use cases.

So r66708 uses a short-lived thread and thread-cache to avoid permanent overhead.

r66712 introduces an experimental new API (rb_nogvl) which allows C-API users to fire some unblock functions in signal handlers. It won't affect grpc's case, but it affects zlib and bignum in core, at least.

It turns out zlib and bignum had the same problem you encountered; but I missed them.

#8 - 01/05/2019 11:45 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN to 2.4: DONTNEED, 2.5: DONTNEED, 2.6: REQUIRED

#9 - 01/07/2019 06:02 PM - dentarg (Patrik Ragnarsson)

After switching to Ruby 2.6.0 for our application on Heroku, we have been getting R12 - Exit timeout errors after each deploy. The application is using Puma in cluster mode. While we haven't tried to reproduce this on Heroku, one of our tests starts our application just to see that it can boot, and we noticed that with 2.6.0, the TERM signal wasn't enough to stop the application. Our test now sends KILL after a while.

I've tried to do a minimal repro of the situation at https://github.com/dentarg/gists/tree/master/gists/ruby-bug-15499. I've sampled the hanged processes (that starts eating ~100% CPU) with Activity Monitor, see the txt files, maybe that can give some clue.

#10 - 01/08/2019 06:22 PM - normalperson (Eric Wong)

> Issue #15499 has been updated by dentarg (Patrik Ragnarsson).

Patrik: can you try r66708? (git 9e66910b3bd85de32e95cf019ed285a36aecfd9e)
It was committed before your comment.

Sorry, I barely have time for Ruby, anymore.

#11 - 01/10/2019 12:10 PM - dentarg (Patrik Ragnarsson)

> Patrik: can you try r66708? (git 9e66910b3bd85de32e95cf019ed285a36aecfd9e)

Eric: Sorry, I should have mentioned it in my comment. I also tried r66716 (ruby 2.7.0dev (2019-01-05 trunk 66716) [x86_64-darwin18]), which can be seen at GitHub. The behaviour was the same as with 2.6.0.

Do you want me to try r66708 specifically and not the commits after? I can do that if so.

#12 - 01/28/2019 06:53 PM - ko1 (Koichi Sasada)

- Description updated

#13 - 01/28/2019 07:20 PM - ko1 (Koichi Sasada)

Sorry I didn't check this ticket (because it seems difficult).

r66708

I'm not sure why another thread is needed. could you explain about it?

r66712

rb_nogvl seems not good name.
Maybe Microsoft will name it rb_thread_call_without_gvl_ex().

I'm not sure the requirement of RB_NOGVL_UBF_ASYNC_SAFE because I can't understand why r66708 is needed.

```
     else if (ubf && vm_living_thread_num(th->vm) == 1) {
-        ubf_th = rb_thread_start_unblock_thread();
+        if (RB_NOGVL_UBF_ASYNC_SAFE) {
+            th->vm->ubf_async_safe = 1;
+        }
+        else {
+            ubf_th = rb_thread_start_unblock_thread();
+        }
     }
```

The condition is always true and maybe r66708 is disabled.
No test?

#14 - 01/28/2019 07:22 PM - ko1 (Koichi Sasada)

I recommend to backport only r66708 for ruby 2.6.1 now.
Honestly speaking I don't understand why it is needed, but eric should know more so I want to believe.

r66712 has several problem so we need discuss more.

#15 - 01/29/2019 05:31 AM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

Applied in changeset ruby_2_6|r66940.

---

merge revision(s) 66708: [Backport #15499]

```
thread.c (call_without_gvl): spawn thread for UBF iff single-threaded

We need another native thread to call some unblocking functions
which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS.  Instead of a
permanent thread in <= 2.5, we can now rely on the thread cache
feature to perform interrupts.

[ruby-core:90865] [Bug #15499]
```

#16 - 01/29/2019 05:36 AM - naruse (Yui NARUSE)

- Backport changed from 2.4: DONTNEED, 2.5: DONTNEED, 2.6: REQUIRED to 2.4: DONTNEED, 2.5: DONTNEED, 2.6: DONE

ruby_2_6 r66940 merged revision(s) 66708.

#17 - 03/26/2020 09:03 AM - naruse (Yui NARUSE)

- Backport changed from 2.4: DONTNEED, 2.5: DONTNEED, 2.6: DONE to 2.4: DONTNEED, 2.5: DONTNEED, 2.6: REQUIRED

#18 - 03/27/2020 02:43 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.4: DONTNEED, 2.5: DONTNEED, 2.6: REQUIRED to 2.4: DONTNEED, 2.5: DONTNEED, 2.6: DONE

The ff98931dac is the fix for 23444302d9 and the 23444302d9 have not been backported into ruby_2_6 branch. 23444302d9 seems a kind of performance improvement.
I will restore Backport field to 2.6: DONE.

#19 - 12/30/2020 11:28 PM - apolcyn (alex polcyn)

Thanks for the followup on this issue. However, was the fix ever released?

I can still reproduce this issue on the following ruby versions, using the grpc gem at version 1.34.0:

```
$ ruby -v
ruby 2.6.6p146 (2020-03-31 revision 67876) [x86_64-linux]

$ ruby -v
ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x86_64-linux]

$ ruby -v
ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x86_64-linux]
```

My system:

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description: Debian GNU/Linux 10 (buster)
Release: 10
Codename: buster
```

Note that this is still NOT an issue on:

```
$ ruby -v
ruby 2.5.8p224 (2020-03-31 revision 67882) [x86_64-linux]
```

#20 - 01/05/2021 02:24 AM - ko1 (Koichi Sasada)

- Status changed from Closed to Assigned

- ruby -v changed from 2.6.0 to master

Thank you for your report. I can reproduce the issue.

```
Thread.new{ Ractor.receive }
Ractor.receive
```

It shouldn't be a Ractor.receive, but the condition are:

- Make two or more threads.
- All threads are waiting for some nogvl operation with custom ubf.

Maybe because it is by 9e66910b3bd85de32e95cf019ed285a36aecfd9e

```
We need another native thread to call some unblocking functions
which aren't RUBY_UBF_IO or RUBY_UBF_PROCESS.  Instead of a
permanent thread in <= 2.5, we can now rely on the thread cache
feature to perform interrupts.
```

But I can't understand what is the "thread cache feature".
I need to investigate more.