Ruby master - Feature #17363

Timeouts

12/03/2020 02:58 PM - marcandre (Marc-Andre Lafortune)

Status: Assigned
Priority: Normal
Assignee: ko1 (Koichi Sasada)
Target version: 

Description

Builtin methods like Queue.pop and Ractor.receive have no timeout parameter.

We should either:

- provide such a parameter
- and/or provide a Timeout::wake that raises an timeout error only if the block is currently sleeping.

Details:

```ruby
q = Queue.new
# ...
elem = Timeout::timeout(42) { q.pop }
# => It is possible that an element is retrieved from the queue but never stored in `elem`

elem = Timeout::wake(42) { q.pop }
# => Guaranteed that either element is retrieved from the queue or an exception is raised, never both
Timeout::wake(42) { loop {} } # => infinite loop
# and/or
elem = q.pop(timeout: 42)
```

Currently, the only reliable way to have a Queue that accepts a timeout is to re-implement it from scratch. This post describe how involved that can be: https://spin.atomicobject.com/2017/06/28/queue-pop-with-timeout-fixed/ 

Related issues:

- Related to Ruby master - Feature #17470: Introduce non-blocking `Timeout.time... Closed
- Related to Ruby master - Feature #17849: Fix Timeout.timeout so that it can b... Open

History

#1 - 12/03/2020 03:11 PM - jeremyevans0 (Jeremy Evans)
I've wanted a timed version of Queue#pop for a long time, to use as the backed for Sequel's connection pool. I was thinking of a separate method (Queue#timed_pop), but a keyword argument works fine too. I think either is better than Timeout.wake.

#2 - 12/03/2020 07:04 PM - Eregon (Benoit Daloze)
+1 for Queue#pop(timeout: 42).

FWIW TruffleRuby already has Queue#receive_timeout as a private method, and this is used to implement Timeout.timeout without creating a new Thread every time.

#3 - 12/04/2020 03:20 AM - nobu (Nobuyoshi Nakada)
I'm positive about that option too.
But I wonder how Timeout.wake works and if it is possible.

#4 - 12/10/2020 08:19 AM - ko1 (Koichi Sasada)
I also positive to introduce timeout but not sure what happens on timeout.
I think timeout: nil is same as no timeout: given. Is it same as other methods?

#5 - 12/10/2020 08:20 AM - ko1 (Koichi Sasada)
Timeout::wake you can implement it with Thread#handle_interrupt(RuntimeError => :never){ ... }.

#6 - 12/10/2020 10:37 AM - marcandre (Marc-Andre Lafortune)
ko1 (Koichi Sasada) wrote in #note-4:

I also positive to introduce timeout but not sure what happens on timeout.
  
  • raise an exception -> which exception?

How about subclassing Timeout::Error to create Queue::Timeout and Ractor::Timeout?
  
  • return nil -> can't recognize returned value

Agreed, it is not a good solution.

I think timeout: nil is same as no timeout: given. Is it same as other methods?

Agree.

I think queue.pop(timeout: 0) should be same as queue.pop(true) but raise Queue:Timeout.
Same idea with Ractor, timeout: 0 is non-blocking version of Ractor.receive/receive_if/select.

#7 - 12/10/2020 11:45 AM - Eregon (Benoit Daloze)
marcandre (Marc-Andre Lafortune) wrote in #note-6:

How about subclassing Timeout::Error to create Queue::Timeout and Ractor::Timeout?

Timeout is stdlib, unlike the other 2 which are in core, so that's an issue.

#8 - 12/10/2020 11:48 AM - marcandre (Marc-Andre Lafortune)
Eregon (Benoit Daloze) wrote in #note-7:

marcandre (Marc-Andre Lafortune) wrote in #note-6:

How about subclassing Timeout::Error to create Queue::Timeout and Ractor::Timeout?

Timeout is stdlib, unlike the other 2 which are in core, so that's an issue.

Good point. We could create Thread::Timeout as a common base class for all 3?

#9 - 12/10/2020 11:53 AM - Eregon (Benoit Daloze)
marcandre (Marc-Andre Lafortune) wrote in #note-8:

Good point. We could create Thread::Timeout as a common base class for all 3?

Thread::TimeoutError then maybe?
Sounds OK, but not sure timeouts are always related to threads (e.g., an IO.select timeout). Might not matter much, so Thread::TimeoutError is fine for me.

#10 - 12/11/2020 05:48 AM - marcandre (Marc-Andre Lafortune)
We could also define ::TimeoutError as base class, and modify timeout lib so that Timeout::Error < ::TimeoutError instead of == as it is currently.

#11 - 12/11/2020 10:35 AM - nobu (Nobuyoshi Nakada)
It is just one line to built-in Timeout::Error.
rb_define_class_under(rb_define_module("Timeout"), "Error", rb_eRuntimeError);

#12 - 12/13/2020 12:29 PM - Eregon (Benoit Daloze)

nobu (Nobuyoshi Nakada) wrote in #note-11:

It is just one line to built-in Timeout::Error.

rb_define_class_under(rb_define_module("Timeout"), "Error", rb_eRuntimeError);

I think that would be confusing, if Timeout::Error is in core, and so a Timeout module is always defined, and yet Timeout.timeout is not defined.

#13 - 12/13/2020 12:30 PM - Eregon (Benoit Daloze)

So another option would be to move the timeout stdlib to core, which could be interesting (can be better optimized, avoid an extra Ruby thread, etc).

#14 - 12/27/2020 04:25 PM - Eregon (Benoit Daloze)

- Related to Feature #17470: Introduce non-blocking 'Timeout.timeout' added

#15 - 01/12/2021 07:31 AM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)
- Status changed from Open to Assigned

#16 - 02/11/2021 11:30 PM - ioquatix (Samuel Williams)

This seems like a good idea. Thank you everyone for the great discussion here.

I agree with the following things:

- Move Timeout to core.
- Add Timeout::Error as base class in core.
- Add new method for predictable timeout during sleeping operations (e.g. Timeout.wake or something similar).

In terms of queue and ractor, I'm less inclined to support:

- timeout: keyword argument.
- Custom exception classes for Ractor, Queue and so on.

I'm not against it, I'm just not sure if it's useful in practice. I think the latter feature should be separate issue/PR if possible.

Finally, I'd also like to suggest that we deprecate Timeout.timeout once this is merged.

#17 - 02/16/2021 01:25 PM - naruse (Yui NARUSE)

Through my experience on implementing write_timeout for net/http, there are 2 layers for this type of APIs. Low-level layer has SELECT (or wait_readable/wait_writable) and nonblock write/read APIs. (IO and Socket)

High-level layer has timeout APIs which is implemented with above. (Net::BufferedIO and Net::HTTP)

#18 - 03/31/2021 06:34 PM - Eregon (Benoit Daloze)

Timeout.wake sounds a bit like Java's Thread.interrupt(), correct?

So it would interrupt blocking calls (File.read/Queue#pop/rb_thread_call_without_gvl/sleep/Mutex#lock/etc) but wouldn't interrupt not-blocking Ruby code like loop[1+1] or while true; 1+1; end.

Also if it happens while the Thread is not doing a blocking call it should probably set a flag that's then checked before any of these blocking calls (like Java's Thread.interrupt()), otherwise it would be too easy to lose such an interrupt/timeout.

I'm not entirely sure how it would work to check the flag just before the blocking call and making sure to not lose an interrupt sent in between, but it should be possible.


#19 - 05/14/2021 06:16 PM - Eregon (Benoit Daloze)

- Related to Feature #17849: Fix Timeout.timeout so that it can be used in threaded Web servers added