## [PATCH] Implement Fiber#raise

10/09/2014 12:29 AM - nome (Knut Franke)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>ioquatix (Samuel Williams)</td>
</tr>
<tr>
<td>Target version</td>
<td>2.7</td>
</tr>
</tbody>
</table>

### Description

While it is possible to implement this in pure Ruby (by wrapping Fiber.yield and Fiber#resume), this feels like a low-level feature that ought to be provided out of the box. Also, the C implementation is more straightforward, and more efficient. Unfortunately, it is not quite possible to implement this as a C extension module (without resorting to wrappers again); cf. the change to make_passing_arg().

#### Example usage:

```ruby
fib = Fiber.new do
  counter = 0
  loop { counter += Fiber.yield }
  counter
end
fib.resume
fib.resume 10
fib.resume 100
fib.raise StopIteration # => 110
```

### Related issues:

- Related to Ruby master - Bug #595: Fiber ignores ensure clause
- Related issues:

### Associated revisions

- **Revision 5fb9d1e1** - 12/28/2018 01:03 PM - samuel
  - Implement Fiber#raise. Fixes #10344.
  - This allows raising exceptions in another fiber, similarly to Thread#raise.
  - `git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66610 b2dd03c8-39d4-4d8f-98ff-823fe69b080e`

- **Revision 66610** - 12/28/2018 01:03 PM - samuel
  - Implement Fiber#raise. Fixes #10344.
  - This allows raising exceptions in another fiber, similarly to Thread#raise.

### History

- **#1** - 10/09/2014 02:58 AM - ko1 (Koichi Sasada)
  - Do you have good example to use it?

- **#2** - 10/09/2014 10:16 AM - funny_falcon (Yura Sokolov)
  - Yes: it is missing piece to make "greenlet" port, in other words - green threads that mimics real threads.
  - To accomplish that there is a need to raise exception in a fiber waiting on event if event fails.
  - As Knut said it could be implemented in a Ruby, but it feels to be dirty way. If implemented in C it will lead to cleaner implementation.

- **#3** - 10/11/2014 04:28 PM - nome (Knut Franke)
  - For some more sophisticated examples, see [https://github.com/nome/coroutines](https://github.com/nome/coroutines). The library does work with vanilla Ruby, but the patch improves performance.
  - Also, similar code can be simplified by using Fiber#raise. Compare e.g. the two implementation of Consumer::Yielder#await at [https://github.com/nome/coroutines/blob/master/lib/coroutines/base.rb](https://github.com/nome/coroutines/blob/master/lib/coroutines/base.rb)

- **#4** - 10/16/2014 04:41 AM - ko1 (Koichi Sasada)
On 2014/10/12 1:28, Knut.Franke@gmx.de wrote:

For some more sophisticated examples, see https://github.com/nome/coroutines. The library does work with vanilla Ruby, but the patch improves performance.

Also, similar code can be simplified by using Fiber#raise. Compare e.g. the two implementation of Consumer::Yielder#await at https://github.com/nome/coroutines/blob/master/lib/coroutines/base.rb

I understand this feature helps some libraries. But I can't understand why it is important.

I'm afraid that introducing such feature increases complexity of Fiber. Basically, I want to recommend strongly that using Fiber as semi-coroutine, restricted feature.

At least, such feature should be located at ext/fiber.

--
// SASADA Koichi at atdot dot net

#5 - 10/18/2014 12:11 PM - funny_falcon (Yura Sokolov)

At least, such feature should be located at ext/fiber.

even if ext/fiber it is still should be implemented in C.

I'm afraid that introducing such feature increases complexity of Fiber.

Not too much. Look at the patch: it uses same hook as rb_fiber_terminate.

By the way, why Thread has this feature? http://www.ruby-doc.org/core-2.1.3/Thread.html#method-i-raise

And why Fiber hasn't? In fact, it is strange that this feature still not presented, cause it is expected to exists.

#6 - 10/18/2014 01:55 PM - nome (Knut Franke)

- File 0001-Implement-Fiber-raise.patch added
- File 0001-Implement-Fiber-raise-in-ext-fiber.patch added

I understand this feature helps some libraries. But I can't understand why it is important.

Without Fiber#raise, libraries need to use some rather inelegant and inefficient workarounds (wrapping or tagging exceptions, wrapping Fiber.yield etc). Also, having Thread#raise arguably leads one to expect the same functionality in Fiber. I'm not sure whether the feature qualifies as "important", but I still think it is worth including.

Basically, I want to recommend strongly that using Fiber as semi-coroutine, restricted feature.

I think Consumer is an example of a semi-coroutine (in the sense that it uses Fiber.yield, not Fiber#transfer) that benefits from having Fiber#raise.

At least, such feature should be located at ext/fiber.

I disagree, because the feature is applicable to the restricted (semi-coroutine) fibers available without requiring 'fiber', and indeed implementable on top of them. Nevertheless, I've attached a variant of the patch which adds the method only in ext/fiber (as a compromise solution).

#7 - 10/23/2014 01:05 AM - ko1 (Koichi Sasada)

Yura Sokolov wrote:

At least, such feature should be located at ext/fiber.

even if ext/fiber it is still should be implemented in C.
I agree.

I'm afraid that introducing such feature increases complexity of Fiber.

Not too much. Look at the patch: it uses same hook as rb_fiber_terminate.

I don't mention about implementation, but specification.

By the way, why Thread has this feature? [http://www.ruby-doc.org/core-2.1.3/Thread.html#method-i-raise](http://www.ruby-doc.org/core-2.1.3/Thread.html#method-i-raise)

And why Fiber hasn't? In fact, it is strange that this feature still not presented, cause it is expected to exists.

Thread#raise is one of inter communication method (but asynchronouse, difficult, unrecommended way).

Fibers can be under control without such communication.

#8 10/23/2014 01:08 AM - ko1 (Koichi Sasada)
Knut Franke wrote:

I understand this feature helps some libraries. But I can't understand why it is important.

Without Fiber#raise, libraries need to use some rather inelegant and inefficient workarounds (wrapping or tagging exceptions, wrapping Fiber.yield etc).

I understand.

Also, having Thread#raise arguably leads one to expect the same functionality in Fiber. I'm not sure whether the feature qualifies as "important", but I still think it is worth including.

I don't expect there is Fiber#raise. Fiber and Thread are different.

Basically, I want to recommend strongly that using Fiber as semi-coutine, restricted feature.

I think Consumer is an example of a semi-coroutine (in the sense that it uses Fiber.yield, not Fiber#transfer) that benefits from having Fiber#raise.

Interesting. I don't know details of this library. Could you explain why it is important in this library?

#9 10/24/2014 04:16 PM - nome (Knut Franke)
Koichi Sasada wrote:

I think Consumer is an example of a semi-coroutine (in the sense that it uses Fiber.yield, not Fiber#transfer) that benefits from having Fiber#raise.

Interesting. I don't know details of this library. Could you explain why it is important in this library?

Consumer provides an abstraction for (semi-)coroutines that accept (consume) values; analogous to an Enumerator, which produces (enumerates) values. Since a consumer may need to do resource cleanup and/or produce a final result, we need some way to signal end of values. I think the most natural way to do this is to raise StopIteration in the consumer: The situation is analogous to calling Enumerator#next when no more values are available, and StopIteration terminates Kernel#loop, which often allows writing consumers very concisely.

In general, I think Fiber is a powerful primitive that can be used by libraries to build abstractions like Enumerator, Consumer and others. While Enumerator does not require the #raise feature, other abstractions can very well benefit from it.

#10 09/19/2017 09:07 AM - yugui (Yuki Sonoda)
- Assignee set to ko1 (Koichi Sasada)
- Target version set to 2.5

Let me add another use case which I discussed with naruse and akr in RubyKaigi.
Fiber\#raise is helpful to improve compatibility of methods with blocks (internal iterators) and Enumerator (external iterators). This happened for me when I tried to write an adapter between two independently-developed APIs.

Here's a simplified example below. You can also find the real example in https://github.com/supership-jp/activerecord-spanner-adapter/blob/master/lib/active_record/connection_adapters/spanner/client.rb.

**Example**

Suppose that we have a third-party API, which we cannot control.

```ruby
def with_transaction
  tx = Transaction.new
  begin
    yield tx
  rescue
    tx.rollback
  else
    tx.commit
  end
end
```

And now I need to begin a transaction in a method and then need to commit or rollback the transaction in other methods.

```ruby
class TransactionManager
  def begin_transaction
    @iter = Transaction.enum_for(:begin_transaction)
    @tx = @iter.next # skip error handlings for simplicity
  end

  def commit_transaction
    loop { @iter.next }
    ensure
      @tx = nil
  end

  def abort_transaction(reason = RuntimeError.new)
    # ??? How can I let the iterator know the error raised
    @iter.raise(reason)
  end
  
  attr_reader :tx
end
```

In other words, internal iterators in Ruby can manage resources in it but external iterators can't even if the user tried to take responsibility of calling cleanup mechanism.

In the real usecase, with_transaction was an API given by google-cloud-spanner gem and the interface of the TransactionManager was the one I had to implement for ActiveRecord.

**Proposal**

Although I could certainly implement the adapter without the native Fiber\#raise, it would be still helpful if internal iterators and external iterators in Ruby were more consistent.

Can we have Fiber\#raise and Enumerator\#raise on top of it to improve the consistency? ko1?

FYI: I've confirmed that the patch by nome still works fine for ruby-trunk.
https://github.com/yugui/ruby/commit/5a04a8b49b5086686fd7f75c6635c95c12ccc6caa8

#11 - 12/01/2017 04:46 PM - naruse (Yui NARUSE)

- Target version deleted (2.5)

#12 - 12/12/2018 01:51 AM - ioquatix (Samuel Williams)

I think this is a good idea. I already ended up implementing it by hand, but it's messy.

https://github.com/socketry/async/blob/e169aac7bc47f251ae7c6ebe94820b411bc2d063f/lib/async/task.rb#L43-L55

#13 - 12/12/2018 01:51 AM - ioquatix (Samuel Williams)

@ko1 (Koichi Sasada) do you think we can aim for 2.6? I don't mind reviewing this patch and merging it if you are happy with it.

#14 - 12/20/2018 01:11 AM - ioquatix (Samuel Williams)
Status changed from Open to Assigned
Assignee changed from ko1 (Koichi Sasada) to ioquatix (Samuel Williams)
Target version set to 2.7

We will experiment and aim to merge this into 2.7

#15 - 12/20/2018 01:11 AM - ioquatix (Samuel Williams)
- Related to Bug #595: Fiber ignores ensure clause added

#16 - 12/28/2018 01:03 PM - Anonymous
- Status changed from Assigned to Closed

Applied in changeset trunk|r66610.

Implement Fiber#raise. Fixes #10344.
This allows raising exceptions in another fiber, similarly to Thread#raise.

#17 - 12/28/2018 01:04 PM - ioquatix (Samuel Williams)
@nome (Knut Franke) can you review the relevant commits and give me feedback?

#18 - 12/30/2018 03:48 PM - naruse (Yui NARUSE)
@ioquatix (Samuel Williams) Is this feature accepted by matz?
A change which is visible from Ruby and non trivial bugs requires matz's approval.

And also you need to add description to NEWS if you add new feature.

#19 - 12/30/2018 05:44 PM - ko1 (Koichi Sasada)
naruse (Yui NARUSE) wrote:

    @ioquatix (Samuel Williams) Is this feature accepted by matz?
    A change which is visible from Ruby and non trivial bugs requires matz's approval.

this is a support comment.

Matz agreed this feature several years ago (maybe I'm only person against, and I also agree to introduce this feature now, except naming issue. but nobody except me has no objection :p)
I'm not sure about naming issue, but he doesn't say anything last time.

Anyway, it is safe to discuss at next dev meeting (I will speak for Samuel).

Thanks,
Koichi

#20 - 12/30/2018 09:56 PM - ioquatix (Samuel Williams)
Thanks I didn't know approval procedure.

I will update NEWS one I'm sure the feature has stabilised.

#21 - 12/31/2018 05:00 AM - naruse (Yui NARUSE)
ko1 (Koichi Sasada) wrote:

    naruse (Yui NARUSE) wrote:

        @ioquatix (Samuel Williams) Is this feature accepted by matz?
        A change which is visible from Ruby and non trivial bugs requires matz's approval.

this is a support comment.

Matz agreed this feature several years ago (maybe I'm only person against, and I also agree to introduce this feature now, except naming issue. but nobody except me has no objection :p)
I'm not sure about naming issue, but he doesn't say anything last time.
Anyway, it is safe to discuss at next dev meeting (I will speak for Samuel).

Sure, thank you for comment.

ioquatix (Samuel Williams) wrote:

Thanks I didn’t know approval procedure.

I will update NEWS one I’m sure the feature has stabilised.

Anyway please update NEWS now to avoid forgetting adding the description.

#22 - 05/28/2019 11:26 AM - decuplet (Nikita Shilnikov)

Shortly after I started to work on a library implementing algebraic effects (https://github.com/dry-rb/dry-effects) I stumbled upon lack of Fiber#raise. From the user POV it is important to signal improper use of effects with a meaningful stacktrace so they can quickly detect what's wrong. It's possible to emulate #raise with checks but it's quite inefficient and seems redundant. Thank you all folks for discussing and working on this, I hope it'll land in 2.7. If you're interested I can show a clear example but the task is almost identical to @ioquatix's one.

#23 - 09/14/2019 09:49 PM - ioquatix (Samuel Williams)

NEWS is updated.

Files

<table>
<thead>
<tr>
<th>Patch Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001-Implement-Fiber-raise.patch</td>
<td>4.12 KB</td>
<td>10/09/2014</td>
<td>nome (Knut Franke)</td>
</tr>
<tr>
<td>0001-Implement-Fiber-raise.patch</td>
<td>3.51 KB</td>
<td>10/18/2014</td>
<td>nome (Knut Franke)</td>
</tr>
<tr>
<td>0001-Implement-Fiber-raise-in-ext-fiber.patch</td>
<td>3.6 KB</td>
<td>10/18/2014</td>
<td>nome (Knut Franke)</td>
</tr>
</tbody>
</table>