Ruby master - Feature #11348

TracePoint API needs events for fiber's switching

07/13/2015 02:01 PM - os97673 (Oleg Sukhodolsky)

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

Description

as discussed in https://github.com/deivid-rodriguez/byebug/issues/153 current implementation of byebug/debase has problem with stepping when Enumarator/Fiber is used. The problem is that Fiber completely replaces stack (w/o any events) while the gems assume that any stack modification has corresponding call/return event.

It would be nice to receive events for Fiber's switching (at least) to be able to track stack for every fiber independently.
Also I think it would be nice to discuss the API before adding it because different set of event will allow to implement different debugger's capabilities.
E.g. plain :fiber-changed (event about every fiber switch) will only allow us to handle every fiber independently (like threads are handled) while debugger's user may expect that step into on enumerator.next will step into enumerator's code and step out from that code should go to the place where enumerator.next has been called.
So, perhaps we should have different events for Fiber#resume and Fiber#yield, but I'm not sure and would like to here other opinions.

Here are two tests which can be used to play with (one for Enumerator, another for Fiber)

```
triangular_numbers = Enumerator.new do |yielder|
  number = 0
  count = 1
  loop do
    number += count
    count += 1
    yielder.yield number
  end
end

print triangular_numbers.next, " "

5.times do
  print triangular_numbers.next, " "
end

fiber = Fiber.new do
  number = 0
  count = 1
  loop do
    number += count
    count += 1
    Fiber.yield number
  end
end

print fiber.resume, " "

5.times do
  print fiber.resume, " "
end
```

Associated revisions

Revision 3af5298e - 08/21/2015 09:51 AM - ko1 (Koichi Sasada)

- include/ruby/ruby.h, cont.c, vm_trace.c: add a new event fiber_switch. We need more discussion about this feature so that I don't write it on NEWS. [Feature #11348]
- test/ruby/test_settracefunc.rb: add tests.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51652 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 51652 - 08/21/2015 09:51 AM - ko1 (Koichi Sasada)

- include/ruby/ruby.h, cont.c, vm_trace.c: add a new event fiber_switch. We need more discussion about this feature so that I don't write it on NEWS. [Feature #11348]
- test/ruby/test_settracefunc.rb: add tests.

Revision 51652 - 08/21/2015 09:51 AM - ko1 (Koichi Sasada)

- include/ruby/ruby.h, cont.c, vm_trace.c: add a new event fiber_switch. We need more discussion about this feature so that I don't write it on NEWS. [Feature #11348]
- test/ruby/test_settracefunc.rb: add tests.

Revision 51652 - 08/21/2015 09:51 AM - ko1 (Koichi Sasada)

- include/ruby/ruby.h, cont.c, vm_trace.c: add a new event fiber_switch. We need more discussion about this feature so that I don't write it on NEWS. [Feature #11348]
- test/ruby/test_settracefunc.rb: add tests.

Revision 51652 - 08/21/2015 09:51 AM - ko1 (Koichi Sasada)

- include/ruby/ruby.h, cont.c, vm_trace.c: add a new event fiber_switch. We need more discussion about this feature so that I don't write it on NEWS. [Feature #11348]
- test/ruby/test_settracefunc.rb: add tests.

Revision 51652 - 08/21/2015 09:51 AM - ko1 (Koichi Sasada)

- include/ruby/ruby.h, cont.c, vm_trace.c: add a new event fiber_switch. We need more discussion about this feature so that I don't write it on NEWS. [Feature #11348]
- test/ruby/test_settracefunc.rb: add tests.

History

#1 - 08/10/2015 01:39 AM - nagachika (Tomoyuki Chikanaga)

- Status changed from Open to Assigned

- Assignee set to ko1 (Koichi Sasada)

#2 - 08/21/2015 09:51 AM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed

Applied in changeset r51652.

- include/ruby/ruby.h, cont.c, vm_trace.c: add a new event fiber_switch. We need more discussion about this feature so that I don't write it on NEWS. [Feature #11348]
- test/ruby/test_settracefunc.rb: add tests.

#3 - 08/21/2015 09:55 AM - ko1 (Koichi Sasada)

- Status changed from Closed to Feedback

I agree to add 'switching' events so I try to commit it.

Could you try it and could you tell me "what" is not enough?

(Actually, I don't increase the number of events)

For example, how about to provide a method to know resume chain like that?

```
Fiber.new{ # f1
  Fiber.new{ # f2
    Fiber.nesting #=> [root_fiber, f1, f2]
  }
}
```

(not sure nesting is good name or not)

#4 - 08/21/2015 12:42 PM - os97673 (Oleg Sukhodolsky)

Koichi Sasada wrote:

> I agree to add 'switching' events so I try to commit it.
>
> Could you try it and could you tell me "what" is not enough?

I'm on vacation right now will play with ruby-head early next week.

#5 - 08/25/2015 01:08 PM - os97673 (Oleg Sukhodolsky)

Oleg Sukhodolsky wrote:

> Koichi Sasada wrote:
>
> > I agree to add 'switching' events so I try to commit it.
> >
> > Could you try it and could you tell me "what" is not enough?

I've played with the event a little bit and here is what I think.
Suppose we have the following program:

```
f1 = Fiber.new do
  loop do
    Fiber.yield 1
  end
end
f1.resume # stopped at the line
f1.resume # want to stop here after step over
f1.resume
f1.resume
```

To implement expected behavior debugger (debase/byebug) store current stack size and stops at next line event with the same (or lower) stack size.
To handle fibers I would expect to use pair (fiber-stack-size, stack-size) and stop only if current fiber stack size is less than original or fiber stack sizes are equal and stack-size is equal or less than original (if you have better idea how to implement this feel free to share it with me).
So to implement this approach we need to detect fiber switch AND figure out if it was "fiber call" or "fiber return" to be able to do this we either need to have two different events, or Fiber.nesting call you propose should help as at the moment we receive "fiber-switch" event, or (worse case for debugger) store the information about the switch and detect if it was call/return at next event we will process. The latter is more complicated for debugger than 1 or 2.

Thus I'd prefer to have fiber-call/fiber-return, or be able to detect direction of the switch at the moment debugger is processing "fiber-switch".
Does this looks reasonable for you? Is it doable on Ruby VM side? Do you have another ideas about implementing the debugger's functionality?

#6 - 09/01/2015 08:57 AM - ko1 (Koichi Sasada)

> f1.resume # stopped at the line
> f1.resume # want to stop here after step over

Only for this purpose, how about to stop every fiber switching? It is true that the program enter newline!

#7 - 09/01/2015 09:12 AM - os97673 (Oleg Sukhodolsky)

Koichi Sasada wrote:

> > f1.resume # stopped at the line
> > f1.resume # want to stop here after step over
>
> Only for this purpose, how about to stop every fiber switching? It is true that the program enter newline!

I'm not sure I've completely understand you suggestion :(Could you please clarify?
I've provided the example ask how to guarantee that we will not stop somewhere inside f1's implementation.