Ruby master - Bug #17529
Ractor Segfaults with GC enabled

01/12/2021 01:06 AM - prajjwal (Prajwal Singh)

Status: Closed
Priority: Normal
Assignee: ko1 (Koichi Sasada)
Target version:
ruby -v: ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x86_64-linux]
Backport: 2.6: DONTNEED, 2.7: DONTNEED, 3.0: DONE

Description
I've been benchmarking Ractor on my machine with the following naive prime number generator:

```ruby
# frozen_string_literal: true
def prime?(n)
  2.upto(n - 1).none? { |i| n % i == 0 }
end
NUM_WORKERS = ARGV[0].to_i
producer = Ractor.new do
  i = 1000000
  loop { Ractor.yield i; i += 1 }
end
workers = (1..NUM_WORKERS).map do
  Ractor.new producer do |producer|
    while n = producer.take
      Ractor.yield [n, prime?(n)]
    end
  end
end
loop do
  _r, { number, prime } = Ractor.select(*workers)
  p number if prime
end
```

The code inevitably segfaults, and it appears to be the garbage collector.

If I stick GC.disable in there, the code happily chugs along for several minutes on end without a problem.

Related issues:
Related to Ruby master - Bug #17489: Ractor segfaults

Associated revisions
Revision a72b7b89 - 03/21/2022 07:52 AM - nagachika (Tomoyuki Chikanaga)
merge revision(s) d0d6227a0da5925acf946a09191f172daf53baf2,fff1edf23ba28267bf57079c269f7fa87530e3fa: [Backport #17529]

```c
**ractor.c** | 13 ++++++++++
1 file changed, 6 insertions(+), 7 deletions(-)
```

Fix Ractor.yield(obj, move: true)
Ractor.yield(obj, move: true) and
Ractor.select(..., yield_value: obj, move: true) tried to yield a value with move semantics, but if the trial is failed, the obj should not become a moved object.

To keep this rule, `wait_moving` wait status is introduced.

New yield/take process:
1. If a ractor tried to yield (move: true), make taking ractor's wait status `wait_moving` and make a moved object by `ractor_move(obj)` and wakeup taking ractor.
2. If a ractor tried to take a message from a ractor waiting for yielding (move: true), wakeup the ractor and wait for (1).

---

bootstraptest/test_ractor.rb | 25 ++++++++++++++++ ractor.c | 73 ++++++++++++++++++++++++++++++++++++--------- ractor_core.h | 1 + 3 files changed, 84 insertions(+), 15 deletions(-)

History
#1 - 01/12/2021 01:12 AM - prajjwal (Prajwal Singh)
- File ractor.crash added

#2 - 01/12/2021 01:42 AM - marcandre (Marc-Andre Lafortune)
- Related to Bug #17489: Ractor segfaults added

#3 - 01/12/2021 01:42 AM - marcandre (Marc-Andre Lafortune)
Thanks for the report.
Probably the same bug as #17489

#4 - 01/12/2021 08:04 AM - ko1 (Koichi Sasada)
I couldn't reproduce it. Could you tell me ARGV[0]?

BTW please fill "ruby -v:" filed with your environment (even if it is in crash log)

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

#5 - 01/12/2021 10:21 AM - prajjwal (Prajwal Singh)

@ko1 (Koichi Sasada)
It crashes for any value of ARGV[0] between 1 and 25 (that I tested).

The fact that its happening so consistently for me and not for you makes me wonder if the problem stems from my version of Linux or GCC? Some other compile time option perhaps?

Here's my GCC version:

Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/lto-wrapper
Target: x86_64-pc-linux-gnu
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 10.2.0 (GCC)

And Linux:

Linux Wraith 5.9.14-arch1-1 #1 SMP PREEMPT Sat, 12 Dec 2020 14:37:12 +0000 x86_64 GNU/Linux
Ruby Configure Args
#6 - 01/12/2021 10:21 AM - prajjwal (Prajwal Singh)
- ruby -v changed from 3.0.0 to ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x86_64-linux]

#7 - 01/12/2021 11:15 AM - prajjwal (Prajwal Singh)
Just confirmed that it only segfaults when ruby is configured with the --enable-shared option (which rbenv does by default).

Even more info:
glibc 2.32-5

#8 - 01/29/2021 08:40 AM - ko1 (Koichi Sasada)
- Status changed from Open to Assigned
- Assignee set to ko1 (Koichi Sasada)

#9 - 02/18/2021 07:26 AM - ko1 (Koichi Sasada)
hmm I can't reproduce it yet. Can someone try it and get more information about it?

#10 - 03/21/2022 12:32 AM - wanabe (_ wanabe)
I confirmed with 3.0.0 that the issue is reproducible.
According to git bisect, it seems to be fixed in fff1ef23ba28267bf57097c269f7fa87530e3fa and d0d6227a0da5925ad946a09191f172da53bafe2.

$ (git checkout origin/ruby_3_0 && git cherry-pick d0d6227a0da5925ad946a09191f172da53bafe2 fff1ef23ba28267bf57097c269f7fa87530e3fa && make miniruby -j8 && ./miniruby -v-W0 segv.rb ruby 3.0.4p197 (2022-03-13 revision b04eb796e4) [x86_64-linux]

$ (git checkout origin/ruby_3_0 && make miniruby -j8) > /dev/null 2>&1 && ./miniruby -v-W0 segv.rb
ruby 3.0.4p197 (2022-03-13 revision f404b21f84) [x86_64-linux]
<internal:ractor>:627: [BUG] Segmentation fault at 0x0000000000000020
ruby 3.0.4p197 (2022-03-13 revision f404b21f84) [x86_64-linux]
-- Control frame information -----------------------------------------------
c:0003 p:0003 s:0015 e:000014 METHOD <internal:ractor>:627

c:0002 p:0019 s:0008 e:000007 BLOCK segv.rb:13 [FINISH]
c:0001 p:---- s:0003 e:000002 (none) [FINISH]

-- Ruby level backtrace information ----------------------------------------
segv.rb:13:in `block (4 levels) in <main>'
<internal:ractor>:627:in `yield'

-- Machine register context ----------------------------------------------
RIP: 0x0000560d0246f8d8 RBP: 0x00007f1c80f28920 RSP: 0x00007f1c80f28800
RAX: 0x0000000000000000 REXX: 0x00007f1c80f28810 RCX: 0x0000000000000000
RDX: 0x0000000000000001 RDI: 0x0000560d04733b98 RSI: 0x0000000000000000
RB: 0x0000560d04733e10 R9: 0x0000000000000000 R10: 0x0000000000000000
R11: 0x0000000000000000 R12: 0x00007f1c80f28820 R13: 0x0000560d04733b70
R14: 0x0000560d047416b8 R15: 0x00007f1c80f28810 EFL: 0x0000000000010246

-- C level backtrace information -----------------------------------------
./miniruby(rb_vm_bugreport+0x4a4) [0x560d02565b34]
./miniruby(rb_bug_for_fatal_signal+0xf4) [0x560d02369a54]
./miniruby(sigsegv+0x4d) [0x560d0224a001d]
[0x7f1c85894520]
./miniruby(reactor_select+0x478) [0x560d0246f8d8]
./miniruby(builtin_inside_class+0x3e) [0x560d0247019e]
./miniruby(vm_exec_core+0x32c2) [0x560d0254d3d1]
./miniruby(rb_vm_exec+0x1a2) [0x560d0254f8d8]
./miniruby(thread_do_start_proc+0x339) [0x560d025057d9]
./miniruby(thread_start_func+0xc84) [0x560d02506554]
./miniruby(thread_start_func+0x0d8e) [0x560d0250682e]
[0x7f1c858e6947]
[0x7f1c8587a4a4]

-- Other runtime information ---------------------------------------------
(snip)

And the following script has been modified to make it easier to try.
1000.times do |q|
  producer = Ractor.new do
    1000.times do |i|
      Ractor.yield true
    end
  end
end

workers = (1..10).map do
  Ractor.new producer do |producer|
    while n = producer.take
      Ractor.yield nil
    end
  rescue Ractor::ClosedError
  end
end

loop do
  _, prime = Ractor.select(*workers)
end

#11 - 03/21/2022 06:47 AM - wanabe (_wanabe)
I guess that the btest failure of ruby_3_0 branch on icc-x64 env may be fixed by git cherry-pick d0d6227a0da5925cadf946a091911f172daf53baf2
fff1edf23ba28267b57097c269f7fa87530e3fa.
(An example of this failure is http://rubyci.s3.amazonaws.com/icc-x64/ruby-3.0/log/20220321T004434Z.log.html.gz#test.rb)

#12 - 03/21/2022 07:10 AM - nagachika (Tomoyuki Chikanaga)
- Backport changed from 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN to 2.6: DONTNEED, 2.7: DONTNEED, 3.0: REQUIRED

#13 - 03/21/2022 07:53 AM - nagachika (Tomoyuki Chikanaga)
- Backport changed from 2.6: DONTNEED, 2.7: DONTNEED, 3.0: REQUIRED to 2.6: DONTNEED, 2.7: DONTNEED, 3.0: DONE

ruby_3_0 a72b7b898c69a116d754d599e8bb061761015255 merged revision(s)
d0d6227a0da5925cadf946a091911f172daf53baf2,fff1edf23ba28267b57097c269f7fa87530e3fa.

#14 - 03/21/2022 07:54 AM - nagachika (Tomoyuki Chikanaga)
- Status changed from Assigned to Closed

Applied in changeset git|a72b7b898c69a116d754d599e8bb061761015255.

merge revision(s) d0d6227a0da5925cadf946a091911f172daf53baf2,fff1edf23ba28267b57097c269f7fa87530e3fa: [Backport #17529]

<table>
<thead>
<tr>
<th>file</th>
<th>change</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ractor.c</td>
<td>13</td>
<td>++-------</td>
</tr>
</tbody>
</table>

alen should be actions number on ractor_select()
alen was number of rs, but it should be actions number
(taking ractors + receiving + yielding).
---
  ractor.c | 13 +++++++---
  1 file changed, 6 insertions(+), 7 deletions(-)

fix Ractor.yield(obj, move: true)
Ractor.yield(obj, move: true) and
Ractor.select(...) yield_value: obj, move: true) tried to yield a
value with move semantics, but if the trial is faild, the obj
should not become a moved object.
To keep this rule, `wait_moving` wait status is introduced.

New yield/take process:
(1) If a ractor tried to yield (move: true), make taking racotr's
  wait status `wait_moving` and make a moved object by
  `ractor_move(obj)` and wake up taking ractor.
(2) If a ractor tried to take a message from a ractor waiting fo
  yielding (move: true), wake up the ractor and wait for (1).
---
bootstrap/test_ractor.rb | 25 +++++++++++++
ra...