Ruby master - Bug #14561

Consistent 2.5.0 seg fault in GC, related to accessing an enumerator in a thread

02/28/2018 11:32 PM - dazuma (Daniel Azuma)

Status:	Closed		
Priority:	Normal		
Assignee:	ioquatix (Samuel Williams)		
Target version:			
ruby -v:	ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-darwin17]	**Backport:**	2.5: DONE

Description

This seg fault happens consistently on OSX (specifically I'm reproing it on a late 2015 Macbook pro running 10.13.3, but it seems to happen on similar machines as well). It happens only on Ruby 2.5.0.

Small repro case:

```
enum = Enumerator.new { |y| y << 1 }
thread = Thread.new { enum.peek }  # enum.next also causes the segfault, but not enum.size
thread.join
GC.start   # <- seg fault here
```

The C-level backtrace identifies this as within the mark phase of GC:

```
-- C level backtrace information -------------------------------------------
0   ruby                                0x000000010f77ced7 rb_vm_bugreport + 135
1   ruby                                0x000000010f602628 rb_bug_context + 472
2   ruby                                0x000000010f6f1491 sigsegv + 81
3   libsystem_platform.dylib            0x00007fff6a779f5a _sigtramp + 26
4   ruby                                0x000000010f61bb93 rb_gc_mark_machine_stack + 99
5   ruby                                0x000000010f76bf39 rb_execution_context_mark + 137
6   ruby                                0x000000010f5ea32b cont_mark + 27
7   ruby                                0x000000010f626a02 gc_marks_rest + 146
8   ruby                                0x000000010f6253c0 gc_start + 2816
9   ruby                                0x000000010f61d628 garbage_collect + 184
10  ruby                                0x000000010f622215 gc_start_internal + 485
11  ruby                                0x000000010f7703be vm_call_cfunc + 286
12  ruby                                0x000000010f759af4 vm_exec_core + 12260
13  ruby                                0x000000010f76ac8e vm_exec + 142
14  ruby                                0x000000010f60c101 ruby_exec_internal + 177
15  ruby                                0x000000010f60bff8 ruby_run_node + 56
16  ruby                                0x000000010f592d1f main + 79

I also ran this against Ruby recompiled with -O0, and got a more detailed backtrace:

-- C level backtrace information -------------------------------------------
0   libruby.2.5.dylib                   0x000000010c416e19 rb_print_backtrace + 25
1   libruby.2.5.dylib                   0x000000010c416f28 rb_vm_bugreport + 136
2   libruby.2.5.dylib                   0x000000010c2096f2 rb_bug_context + 450
3   libruby.2.5.dylib                   0x000000010c35b4ee sigsegv + 94
4   libsystem_platform.dylib            0x00007fff6a779f5a _sigtramp + 26
5   libruby.2.5.dylib                   0x000000010c2395a1 mark_locations_array + 49
6   libruby.2.5.dylib                   0x000000010c22a5bb gc_mark_locations + 75
7   libruby.2.5.dylib                   0x000000010c22a7d9 mark_stack_locations + 41
8   libruby.2.5.dylib                   0x000000010c22a79f rb_gc_mark_machine_stack + 79
9   libruby.2.5.dylib                   0x000000010c3f8868 rb_execution_context_mark + 264
10  libruby.2.5.dylib                   0x000000010c1e263e cont_mark + 46
11  libruby.2.5.dylib                   0x000000010c1e2572 fiber_mark + 146
12  libruby.2.5.dylib                   0x000000010c22f4c6 gc_mark_children + 1094
13  libruby.2.5.dylib                   0x000000010c23734c gc_mark_stacked_objects + 108
14  libruby.2.5.dylib                   0x000000010c237a5b gc_mark_stacked_objects_all + 27
15  libruby.2.5.dylib                   0x000000010c236cb1 gc_marks_rest + 129
16  libruby.2.5.dylib                   0x000000010c238787 gc_marks + 103
```

```
17  libruby.2.5.dylib                  0x000000010c2352e2 gc_start + 802
18  libruby.2.5.dylib                  0x000000010c22ca18 garbage_collect + 56
19  libruby.2.5.dylib                  0x000000010c231f7d gc_start_internal + 493
20  libruby.2.5.dylib                  0x000000010c401f2a call_cfunc_m1 + 42
21  libruby.2.5.dylib                  0x000000010c400d1d vm_call_cfunc_with_frame + 605
22  libruby.2.5.dylib                  0x000000010c3fc41d vm_call_cfunc + 173
23  libruby.2.5.dylib                  0x000000010c3fb8fe vm_call_method_each_type + 190
24  libruby.2.5.dylib                  0x000000010c3fb690 vm_call_method + 160
25  libruby.2.5.dylib                  0x000000010c3fb5e5 vm_call_general + 53
26  libruby.2.5.dylib                  0x000000010c3e784e vm_exec_core + 8974
27  libruby.2.5.dylib                  0x000000010c3f6fe6 vm_exec + 182
28  libruby.2.5.dylib                  0x000000010c3f7d5b rb_iseq_eval_main + 43
29  libruby.2.5.dylib                  0x000000010c214208 ruby_exec_internal + 232
30  libruby.2.5.dylib                  0x000000010c214111 ruby_exec_node + 33
31  libruby.2.5.dylib                  0x000000010c2140d0 ruby_run_node + 64
32  ruby                               0x000000010c16ff2f main + 95
```

As far as I can tell, the C instruction triggering the segfault is here in gc.c (around line 4064):

```
static void
mark_locations_array(rb_objspace_t *objspace, register const VALUE *x, register long n)
{
    VALUE v;
    while (n--) {
        v = *x;             // <----- Seems to be crashing here?
        gc_mark_maybe(objspace, v);
        x++;
    }
}
```

Indicating a bad pointer in the machine stack.

I'm not sufficiently familiar with the VM internals to make much further progress, but I hope the repro case is helpful. It seems to require accessing an Enumerator element within a separate thread, and then waiting for the thread to end.

Related issues:

Related to Ruby master - Bug #14714: Ruby 2.5.1 Segmentation Fault in GC	**Closed**
Related to Ruby master - Bug #14334: Segmentation fault after running rspec (...	**Closed**
Is duplicate of Ruby master - Bug #15362: [PATCH] Avoid GCing dead stack afte...	**Closed**
Has duplicate Ruby master - Bug #15308: SegFault in GC under Ruby 2.5.3 on OS X	**Closed**
Has duplicate Ruby master - Bug #15221: Segfault in Ruby VM	**Closed**

Associated revisions

Revision 66111 - 12/01/2018 03:49 AM - samuel

Avoid GCing dead stack after switching away from a fiber

Fixes https://bugs.ruby-lang.org/issues/14561 and discussed
https://bugs.ruby-lang.org/issues/15362.

Revision 0e0d0c47 - 01/10/2019 02:18 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 66111: [Backport #14561]

```
    Avoid GCing dead stack after switching away from a fiber

    Fixes <https://bugs.ruby-lang.org/issues/14561> and discussed
    <https://bugs.ruby-lang.org/issues/15362>.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@66777 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66777 - 01/10/2019 02:18 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 66111: [Backport #14561]

```
Avoid GCing dead stack after switching away from a fiber

Fixes <https://bugs.ruby-lang.org/issues/14561> and discussed
<https://bugs.ruby-lang.org/issues/15362>.
```

History

#1 - 03/01/2018 03:09 AM - ko1 (Koichi Sasada)

I can't reproduce this issue with current trunk (2.6) on Linux / Windows.
clang (mac) issue?

#2 - 03/01/2018 05:28 AM - dazuma (Daniel Azuma)

Yes, I have been able to reproduce this issue only on mac.

#3 - 03/01/2018 06:04 AM - harbirg (Harbir G)

I can also reproduce the same crash on MacOS High Sierra 10.13.2. Occurs on both 2.5.0 and 2.6.0-preview1.

#4 - 03/01/2018 10:44 AM - ned (Ned Hadzo)

I can reproduce, MacOS High Sierra 10.13.3 (17D47)

```
nedims-MacBook-Pro:~ nedim$ irb
2.5.0 :001 > enum = Enumerator.new { |y| y << 1 }
 => #<Enumerator: #<Enumerator::Generator:0x00007fe043133b90>:each>
2.5.0 :002 > thread = Thread.new { enum.peek } # enum.next also causes the segfault, but not enum.size
 => #<Thread:0x00007fe0431e7640@(irb):2 run>
2.5.0 :003 > thread.join
 => #<Thread:0x00007fe0431e7640@(irb):2 dead>
2.5.0 :004 > GC.start
(irb):4: [BUG] Segmentation fault at 0x0000700006d0fbc0
ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-darwin17]
```

#5 - 03/01/2018 03:56 PM - stuarthadfield (Stuart Hadfield)

harbirg (Harbir G) wrote:

> I can also reproduce the same crash on MacOS High Sierra 10.13.2. Occurs on both 2.5.0 and 2.6.0-preview1.

+1. Reliably reproducible on Mac, ruby 2.5.0 OSX El Capitan 10.11.6

Currently causing unit tests to seg fault on production code. Does not occur on Circle Env which is Linux based.

#6 - 03/02/2018 05:31 AM - nobu (Nobuyoshi Nakada)

- Description updated

Seems marking dead fiber's stack.

#7 - 03/11/2018 07:38 PM - brian-kephart (Brian Kephart)

I think I'm running into the same bug. I'm new to reading these types of traces, so please let me know if this needs to be a separate issue instead.
Running Ruby 2.5.0 on OS X.

```
/Users/briankephart/.rvm/rubies/ruby-2.5.0/lib/ruby/2.5.0/socket.rb:227: [BUG] Segmentation fault at 0x0000000
10dd5fa3a
ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-darwin17]

-- Crash Report log information --------------------------------------------
   See Crash Report log file under the one of following:
     * ~/Library/Logs/DiagnosticReports
     * /Library/Logs/DiagnosticReports
   for more details.
Don't forget to include the above Crash Report log file in bug reports.

-- Control frame information -----------------------------------------------
c:0014 p:---- s:0136 e:000135 CFUNC  :getaddrinfo
c:0013 p:0034 s:0126 e:000125 METHOD /Users/briankephart/.rvm/rubies/ruby-2.5.0/lib/ruby/2.5.0/socket.rb:227
c:0012 p:0073 s:0115 e:000114 METHOD /Users/briankephart/.rvm/rubies/ruby-2.5.0/lib/ruby/2.5.0/socket.rb:631
c:0011 p:0034 s:0102 e:000101 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/webpacker-3.2.2/lib/webpack
er/dev_server.rb:14
c:0010 p:0032 s:0098 e:000097 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/webpacker-3.2.2/lib/webpack
er/dev_server_proxy.rb:7
c:0009 p:0014 s:0090 E:001848 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/rack-proxy-0.6.3/lib/rack/p
roxy.rb:57
c:0008 p:0041 s:0085 E:0018d8 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/skylight-core-2.0.0.beta2/l
ib/skylight/core/probes/middleware.rb:28
c:0007 p:0020 s:0074 E:001940 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/railties-5.2.0.rc1/lib/rail
```

```
s/engine.rb:524
c:0006 p:0026 s:0068 E:001998 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/config
uration.rb:225
c:0005 p:0258 s:0063 E:001a98 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/server
.rb:624
c:0004 p:0026 s:0038 E:002590 METHOD /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/server
.rb:438
c:0003 p:0065 s:0026 E:0025e8 BLOCK  /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/server
.rb:302 [FINISH]
c:0002 p:0125 s:0016 E:002690 BLOCK  /Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/thread
_pool.rb:120 [FINISH]
c:0001 p:---- s:0003 e:000002 (none) [FINISH]

-- Ruby level backtrace information ----------------------------------------
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/thread_pool.rb:120:in `block in spawn_threa
d'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/server.rb:302:in `block in run'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/server.rb:438:in `process_client'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/server.rb:624:in `handle_request'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/puma-3.11.2/lib/puma/configuration.rb:225:in `call'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/railties-5.2.0.rc1/lib/rails/engine.rb:524:in `call'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/skylight-core-2.0.0.beta2/lib/skylight/core/probes/middleware.rb
:28:in `call'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/rack-proxy-0.6.3/lib/rack/proxy.rb:57:in `call'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/webpacker-3.2.2/lib/webpacker/dev_server_proxy.rb:7:in `rewrite_
response'
/Users/briankephart/.rvm/gems/ruby-2.5.0/gems/webpacker-3.2.2/lib/webpacker/dev_server.rb:14:in `running?'
/Users/briankephart/.rvm/rubies/ruby-2.5.0/lib/ruby/2.5.0/socket.rb:631:in `tcp'
/Users/briankephart/.rvm/rubies/ruby-2.5.0/lib/ruby/2.5.0/socket.rb:227:in `foreach'
/Users/briankephart/.rvm/rubies/ruby-2.5.0/lib/ruby/2.5.0/socket.rb:227:in `getaddrinfo'

-- Machine register context ------------------------------------------------
 rax: 0x0000000000000000 rbx: 0x00007fc5eef71ec0 rcx: 0x0000000000010000
 rdx: 0x000070000c3b3c80 rdi: 0x000000010dd5fa38 rsi: 0x00007fc5eef71ec0
 rbp: 0x000070000c3b3c70 rsp: 0x000070000c3b3c38  r8: 0x0000000000000000
  r9: 0xffffffff00000000 r10: 0x000000010d7fa738 r11: 0xfffff0009ac08e64
 r12: 0x00007fffaa5a5f40 r13: 0x00007fff717f2864 r14: 0x000070000c3b3c80
 r15: 0x00007fc5eef71ed0 rip: 0x00007fff717f2868 rfl: 0x0000000000010206

-- C level backtrace information -------------------------------------------
0   libruby.2.5.dylib                   0x000000010d9ebd07 rb_vm_bugreport + 135
1   libruby.2.5.dylib                   0x000000010d870978 rb_bug_context + 472
2   libruby.2.5.dylib                   0x000000010d960151 sigsegv + 81
3   libsystem_platform.dylib            0x00007fff717c6f5a _sigtramp + 26
4   libsystem_trace.dylib               0x00007fff717f2868 _os_log_cmp_key + 4

-- Other runtime information -----------------------------------------------

* Loaded script: puma: cluster worker 0: 2204 [brian-becca]

* Loaded features:

    0 enumerator.so
    1 thread.rb
    2 rational.so
    3 complex.so
    ... lots more
```

#8 - 03/14/2018 11:40 AM - ioquatix (Samuel Williams)

- File dump.txt added

- File ruby_2018-03-14-222035_Fukurou.crash added

- File ruby_2018-03-14-205753_Fukurou.crash added

I believe I've run into this bug too.

macOS: 10.13.3
Ruby: 2.5.0

I was running this code:

https://github.com/socketry/async-websocket/tree/ruby-segv

Specifically, in the "chat/" directory, run ./config.ru to start the server, and then run client.rb several times. Finally, press Ctrl-C on the server. Sometimes it crash. Sometimes it's okay. I don't believe I experienced this with previous version of Ruby.

I can help reproduce the issue if needed.

#9 - 03/16/2018 08:23 PM - wanabe (_ wanabe)

git bisect shows this is from r60440 [Feature #14038].

#10 - 03/18/2018 01:42 PM - wanabe (_ wanabe)

It seems to be a FIBER_USE_NATIVE == 0 environment issue.
Perhaps it may be a potential Enumerator's behaviour issue.

First, Fiber.new and Fiber#resume must be in same thread, but Enumerator.new and Enumerator#peek don't have to be.
Because Fiber.new calls fiber_t_alloc() immediately, but Enumerator.new doesn't. He is lazy :)
So Enumerator can take out machine stack value of killed-thread.
I think Thread.new { enum.peek } should raise FiberError.
But it is big incompatibility and not realistic.

There is no "marking dead fiber's stack" problem on FIBER_USE_NATIVE != 0 environment.
Because fiber_setcontext() set oldfib->cont.saved_ec.machine.stack_end = NULL; and skip machine stack mark when ec->machine.stack_end == NULL in rb_execution_context_mark().

There are some ways:

1. fiber_mark checks not only fib->status but also fib->cont->saved_ec.thread_ptr->status on FIBER_USE_NATIVE == 0 environment.
2. thread_cleanup_func() makes all fibers FIBER_TERMINATED.
3. etc.

#11 - 04/21/2018 01:09 PM - ioquatix (Samuel Williams)

Thanks so much for your effort to isolate this issue.

I don't think any of my code is violating "First, Fiber.new and Fiber#resume must be in same thread, but Enumerator.new and Enumerator#peek don't have to be."

I will check.

#12 - 05/02/2018 12:56 PM - ioquatix (Samuel Williams)

I don't know if this is related or not, but I just now saw a very odd error.

```
Traceback (most recent call last):
    7: from /Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-1.8.0/lib/async/task.rb:74:in `block in initialize'
    6: from /Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-io-1.10.0/lib/async/io/socket.rb:83:in `block in acc
ept'
    5: from /Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-io-1.10.0/lib/async/io/socket.rb:83:in `ensure in bl
ock in accept'
    4: from /Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-1.8.0/lib/async/wrapper.rb:135:in `close'
    3: from /Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-1.8.0/lib/async/wrapper.rb:186:in `cancel_monitor'
    2: from /Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-1.8.0/lib/async/wrapper.rb:186:in `close'
    1: from /Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-1.8.0/lib/async/wrapper.rb:186:in `deregister'
/Users/samuel/.rvm/gems/ruby-2.5.0/gems/async-1.8.0/lib/async/wrapper.rb:186:in `lock': deadlock; recursive lo
cking (ThreadError)
```

Notice the last four lines all point to the same line, and yet have different method names.

The line in question is this one: https://github.com/socketry/async/blob/v1.8.0/lib/async/wrapper.rb#L186

The only method named deregister is here: https://github.com/socketry/async/blob/v1.8.0/lib/async/debug/selector.rb#L52

Even thought there was such an error the spec passed.

I don't know how such a thing could happen. It seems like corruption in the VM.

#13 - 05/02/2018 12:57 PM - ioquatix (Samuel Williams)

By the way, it only happened once. Re-running the same spec several more times didn't generate any further odd output.

#14 - 11/30/2018 05:18 AM - aselder (Andrew Selder)

Would it possible to get this addressed? It blocking my entire organization from upgrading past Ruby 2.4

It's reproducible on all 2.5 releases as well as the 2.6 preview releases.

It's been reported multiple times:
https://bugs.ruby-lang.org/issues/14334
https://bugs.ruby-lang.org/issues/14561
https://bugs.ruby-lang.org/issues/14714
https://bugs.ruby-lang.org/issues/15308

Our friend wanabe has even found the commit that introduced the error. I'd love to help out and try and solve this, but I'm afraid I'd just screw things up in the GC. Please let me know if there is anything I can do to help get a fix out.

#15 - 11/30/2018 05:20 AM - ioquatix (Samuel Williams)

I thought this was fixed already in 2.6 - but obviously not. That's bad :(

#16 - 11/30/2018 08:27 PM - alanwu (Alan Wu)

I have a patch for this, #15362 if anyone could take a look.

#17 - 12/01/2018 06:52 AM - ioquatix (Samuel Williams)

- Backport deleted (2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN)

- Assignee set to ioquatix (Samuel Williams)

- Status changed from Open to Closed

Thanks alanwu (Alan Wu) this has been applied to trunk and should be back-ported to 2.5 shortly as per #15362. As the supplied spec is passing, I'm going to assume this is now fixed. Feel free to open a new bug report if there are further issues. Thanks everyone for your time and effort.

#18 - 12/10/2018 03:28 AM - nagachika (Tomoyuki Chikanaga)

- Backport set to 2.5: REQUIRED

#19 - 01/10/2019 07:26 AM - nagachika (Tomoyuki Chikanaga)

- Is duplicate of Bug #15362: [PATCH] Avoid GCing dead stack after switching away from a fiber added

#20 - 01/10/2019 02:18 PM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.5: REQUIRED to 2.5: DONE

ruby_2_5 r66777 merged revision(s) 66111.

#21 - 02/14/2019 12:36 AM - wanabe (_ wanabe)

- Related to Bug #14714: Ruby 2.5.1 Segmentation Fault in GC added

#22 - 06/15/2019 12:11 AM - jeremyevans0 (Jeremy Evans)

- Has duplicate Bug #15308: SegFault in GC under Ruby 2.5.3 on OS X added

#23 - 06/15/2019 12:17 AM - jeremyevans0 (Jeremy Evans)

- Has duplicate Bug #15221: Segfault in Ruby VM added

#24 - 08/26/2019 04:46 PM - jeremyevans0 (Jeremy Evans)

- Related to Bug #14334: Segmentation fault after running rspec (ruby/2.5.0/erb.rb:885 / simplecov/source_file.rb:85) added

Files

ruby_2018-03-14-222035_Fukurou.crash	38.6 KB	03/14/2018	ioquatix (Samuel Williams)
ruby_2018-03-14-205753_Fukurou.crash	38.6 KB	03/14/2018	ioquatix (Samuel Williams)
dump.txt	51.4 KB	03/14/2018	ioquatix (Samuel Williams)