Ruby - Bug #16682

Ruby 2.7.0p0 crash on exit if there is an active RUBY_INTERNAL_EVENT_GC_EXIT tracepoint

03/09/2020 11:27 AM - byroot (Jean Boussier)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-darwin19]	**Backport:**	2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONE

Description

```
[BUG] object allocation during garbage collection phase
ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-darwin19]

-- 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:0001 p:0001 s:0003 E:002690 (none) [FINISH]


-- C level backtrace information -------------------------------------------
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_vm_bugreport+0x96) [0x10fa9f266]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_bug+0xcc) [0x10faabb86]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(newobj_slowpath+0x99c) [0x10f8f939c]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(newobj_slowpath_wb_protected+0x14) [0x10f8f89d4]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_str_buf_new+0x1e) [0x10fa151be]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_enc_vsprintf+0x48) [0x10fa03178]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_vraise+0x14) [0x10f8d4d84]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_raise+0x7b) [0x10f8d052b]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_check_typeddata+0xf3) [0x10f8d19a3]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(tp_call_trace+0x2a) [0x10faa1aaa]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_exec_event_hooks+0x163) [0x10faa0ab3]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_objspace_call_finalizer+0x8b7) [0x10f8eded7]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(rb_ec_cleanup+0x2de) [0x10f8dc35e]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(ruby_run_node+0x5f) [0x10f8dc4ff]
/Users/byroot/.rubies/ruby-2.7.0/bin/ruby(main+0x5d) [0x10f833d0d]
```

It also crash in other circumstances, but I'm not able to reproduce them as easily.

Older versions are not affected.

I created a repository to easily reproduce the issue: https://github.com/casperisfine/ruby-tracepoint-crash

Associated revisions

Revision b385f7670ffa420790bc548747fa4b58c4c5d8f6 - 03/30/2020 03:41 AM - alanwu (Alan Wu)

Clear all trace events during teardown

Since 0c2d81dada, not all trace events are cleared during VM teardown.
This causes a crash when there is a tracepoint for
RUBY_INTERNAL_EVENT_GC_EXIT active during teardown.

The commit looks like a refactoring commit so I think this change was
unintentional.

[Bug #16682]

Revision b385f7670ffa420790bc548747fa4b58c4c5d8f6 - 03/30/2020 03:41 AM - alanwu (Alan Wu)

Clear all trace events during teardown

Since 0c2d81dada, not all trace events are cleared during VM teardown.
This causes a crash when there is a tracepoint for
RUBY_INTERNAL_EVENT_GC_EXIT active during teardown.

The commit looks like a refactoring commit so I think this change was
unintentional.

[Bug #16682]

Revision b385f767 - 03/30/2020 03:41 AM - alanwu (Alan Wu)

Clear all trace events during teardown

Since 0c2d81dada, not all trace events are cleared during VM teardown.
This causes a crash when there is a tracepoint for
RUBY_INTERNAL_EVENT_GC_EXIT active during teardown.

The commit looks like a refactoring commit so I think this change was
unintentional.

[Bug #16682]

Revision b5fa156b7907b8ea1baf8b9f0cb6a66b0fecb3d4 - 03/30/2020 10:15 AM - alanwu (Alan Wu)

Clear all trace events during teardown

Since 0c2d81dada, not all trace events are cleared during VM teardown.
This causes a crash when there is a tracepoint for
RUBY_INTERNAL_EVENT_GC_EXIT active during teardown.

The commit looks like a refactoring commit so I think this change was
unintentional.

[Bug #16682]

(cherry picked from commit b385f7670ffa420790bc548747fa4b58c4c5d8f6)

Revision b5fa156b7907b8ea1baf8b9f0cb6a66b0fecb3d4 - 03/30/2020 10:15 AM - alanwu (Alan Wu)

Clear all trace events during teardown

Since 0c2d81dada, not all trace events are cleared during VM teardown.
This causes a crash when there is a tracepoint for
RUBY_INTERNAL_EVENT_GC_EXIT active during teardown.

The commit looks like a refactoring commit so I think this change was
unintentional.

[Bug #16682]

(cherry picked from commit b385f7670ffa420790bc548747fa4b58c4c5d8f6)

Revision b5fa156b - 03/30/2020 10:15 AM - alanwu (Alan Wu)

Clear all trace events during teardown

Since 0c2d81dada, not all trace events are cleared during VM teardown.
This causes a crash when there is a tracepoint for
RUBY_INTERNAL_EVENT_GC_EXIT active during teardown.

The commit looks like a refactoring commit so I think this change was
unintentional.

[Bug #16682]

(cherry picked from commit b385f7670ffa420790bc548747fa4b58c4c5d8f6)

History

#1 - 03/09/2020 12:29 PM - byroot (Jean Boussier)

I patched my ruby to print the exception message before it tries to allocate, and ran it a few times:

```
.wrong argument type 140351662971400
                                    (expected tracepoint)

.wrong argument type 140321405262280
                                    (expected tracepoint)

.wrong argument type 140682039908840
                                    (expected tracepoint)

.wrong argument type 140494755846600
                                    (expected tracepoint)

.wrong argument type 140445212727760
                                    (expected tracepoint)

.wrong argument type 140449339922920
                                    (expected tracepoint)
```

#2 - 03/09/2020 12:50 PM - methodmissing (Lourens Naudé)

The tracepoint instance appears to not be of type TypedData anymore, fails the type check, exception is raised, which allocs a String for the message.

```
#15 0x000055935811e737 in tpptr (tpval=94091343800880) at vm_trace.c:771
771     TypedData_Get_Struct(tpval, rb_tp_t, &tp_data_type, tp);
(gdb) p ((((struct RBasic*)(tpval))->flags & RUBY_T_MASK) == RUBY_T_DATA)
$21 = 0
(gdb) p ((struct RTypedData *)(tpval))->typed_flag
$22 = 0
(gdb) p tp_data_type
$23 = {wrap_struct_name = 0x5593582028d0 "tracepoint", function = {dmark = 0x55935811e167 <tp_mark>, dfree = 0
x0,
    dsize = 0x55935811e1c3 <tp_memsize>, dcompact = 0x0, reserved = {0x0}}, parent = 0x0, data = 0x0, flags =
1}
```

byroot (Jean Boussier) wrote in #note-1:

> I patched my ruby to print the exception message before it tries to allocate, and ran it a few times:
>
> ```
> .wrong argument type 140351662971400
>                                     (expected tracepoint)
>
> .wrong argument type 140321405262280
>                                     (expected tracepoint)
>
> .wrong argument type 140682039908840
>                                     (expected tracepoint)
>
> .wrong argument type 140494755846600
>                                     (expected tracepoint)
>
> .wrong argument type 140445212727760
>                                     (expected tracepoint)
>
> .wrong argument type 140449339922920
>                                     (expected tracepoint)
> ```

#3 - 03/09/2020 12:58 PM - byroot (Jean Boussier)

> fails the type check, exception is raised, which allocs a String for the message.

Yep, the allocation during garbage collection phase is merely a fallout following a previous issue.

So I patched my ruby a bit further to get the actual class name:

```
diff --git a/error.c b/error.c
index b9ec8427e652..5cb57057bced 100644
--- a/error.c
+++ b/error.c
@@ -910,9 +910,13 @@ rb_check_typeddata(VALUE obj, const rb_data_type_t *data_type)
```

```
       wrong_type:
  etype = builtin_class_name(obj);
  if (!etype)
+                 printf("wrong_type: wrong argument type %s (expected %s)",
+        RSTRING_PTR(rb_mod_name(rb_obj_class(obj))), data_type->wrap_struct_name);
       rb_raise(rb_eTypeError, "wrong argument type %"PRIsVALUE" (expected %s)",
         rb_obj_class(obj), data_type->wrap_struct_name);
       wrong_datatype:
```

Gives me:

```
.wrong_type: wrong argument type TracePoint (expected tracepoint)
```

I need to dig a bit deeper, to understand the difference though.

#4 - 03/09/2020 03:53 PM - byroot (Jean Boussier)

So apparently the TracePoint instance type is set to T_NONE. So I presume it means it was collected?

#5 - 03/10/2020 03:29 PM - ko1 (Koichi Sasada)

how about to disable this TP?

#6 - 03/10/2020 04:14 PM - byroot (Jean Boussier)

ko1 (Koichi Sasada) wrote in #note-5:

> how about to disable this TP?

at_exit { tracepoint.disable } indeed prevent the bug from firing in my reproduction repo, and I've used this workaround a while ago successfully.

But it's merely hiding the bug not fixing it. Our application is segfauling a lot on 2.7.0p0, and not just during the exit sequence, so I suspect this can also happen after a regular GC run.

#7 - 03/10/2020 04:32 PM - methodmissing (Lourens Naudé)

byroot (Jean Boussier) wrote in #note-6:

> ko1 (Koichi Sasada) wrote in #note-5:
>
> > how about to disable this TP?
>
> at_exit { tracepoint.disable } indeed prevent the bug from firing in my reproduction repo, and I've used this workaround a while ago successfully.
>
> But it's merely hiding the bug not fixing it. Our application is segfauling a lot on 2.7.0p0, and not just during the exit sequence, so I suspect this can also happen after a regular GC run.

A cleaner workaround (and yes I agree this doesn't fix the segfault) would be to define a finalizer on the tracepoint instead, which disables it.

The downside of NOT emitting the tracepoint for GC_EXIT could be undefined behaviour for profilers or users of the Tracepoint API where the GC_ENTER event is not paired with a GC_EXIT one anymore. Although this may not matter on shutdown as the process is exiting anyways. Filtering tracepoint objects of type T_NONE would also result in this behaviour.

Jean, the crashes references other TP events too, or just the GC ones?

#8 - 03/10/2020 04:41 PM - byroot (Jean Boussier)

> A cleaner workaround (and yes I agree this doesn't fix the segfault) would be to define a finalizer on the tracepoint instead, which disables it.

I tried that without success.

> The downside of NOT emitting the tracepoint for GC_EXIT could be undefined behaviour for profilers or users of the Tracepoint API

From my tests on 2.6.x, it seems that previously an active TracePoint would simply not be GCed at all.

> the crashes references other TP events too, or just the GC ones?

To clarify here's a sample of the other segfault / crash I'm witnessing and that I suspect are related (sorry, it doesn't print the C level backtrace info):

```
/tmp/bundle/ruby/2.7.0/bundler/gems/rails-40b7d93c5bf0/activemodel/lib/active_model/attribute_methods.rb:377:
[BUG] try to mark T_NONE object
ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0105 p:---- s:0585 e:000584 CFUNC  :module_eval
c:0104 p:0226 s:0578 e:000577 METHOD /tmp/bundle/ruby/2.7.0/bundler/gems/rails-40b7d93c5bf0/activemodel/lib/ac
tive_model/attribute_methods.rb:377
c:0103 p:0031 s:0567 e:000566 BLOCK  /tmp/bundle/ruby/2.7.0/bundler/gems/rails-40b7d93c5bf0/activemodel/lib/ac
tive_model/attribute_methods.rb:213 [FINISH]
c:0102 p:---- s:0561 e:000560 CFUNC  :each
c:0101 p:0025 s:0557 e:000556 METHOD /tmp/bundle/ruby/2.7.0/bundler/gems/rails-40b7d93c5bf0/activemodel/lib/ac
tive_model/attribute_methods.rb:210


tmp/bundle/ruby/2.7.0/gems/bootsnap-1.4.6/lib/bootsnap/compile_cache/iseq.rb:19: [BUG] try to mark T_NONE obje
ct
ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0105 p:---- s:0589 e:000588 CFUNC  :load_from_binary
c:0104 p:0017 s:0584 e:000583 METHOD /tmp/bundle/ruby/2.7.0/gems/bootsnap-1.4.6/lib/bootsnap/compile_cache/ise
q.rb:19 [FINISH]
c:0103 p:---- s:0578 e:000577 CFUNC  :fetch
c:0102 p:0089 s:0571 e:000570 METHOD /tmp/bundle/ruby/2.7.0/gems/bootsnap-1.4.6/lib/bootsnap/compile_cache/ise
q.rb:38 [FINISH]
c:0101 p:---- s:0565 e:000564 CFUNC  :require
c:0100 p:0035 s:0560 e:000559 METHOD /tmp/bundle/ruby/2.7.0/gems/zeitwerk-2.3.0/lib/zeitwerk/kernel.rb:16 [FIN
ISH]


/tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/simple.rb:100: [BUG] Segmentation fault at 0x000000000
0000020
ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0230 p:0069 s:1458 e:001455 BLOCK  /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/simple.rb:100 [FI
NISH]
c:0229 p:---- s:1451 e:001450 IFUNC
c:0228 p:---- s:1448 e:001447 CFUNC  :each
c:0227 p:---- s:1445 e:001444 CFUNC  :inject
c:0226 p:0057 s:1440 e:001439 METHOD /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/simple.rb:93
c:0225 p:0007 s:1431 e:001430 METHOD /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/base.rb:68
c:0224 p:0009 s:1425 e:001424 BLOCK  /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/fallbacks.rb:74 [
FINISH]
c:0223 p:---- s:1421 e:001420 CFUNC  :each


/tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/simple.rb:100: [BUG] Segmentation fault at 0x000000000
0000010
ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0079 p:0069 s:0425 e:000422 BLOCK  /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/simple.rb:100 [FI
NISH]
c:0078 p:---- s:0418 e:000417 IFUNC
c:0077 p:---- s:0415 e:000414 CFUNC  :each
c:0076 p:---- s:0412 e:000411 CFUNC  :inject
c:0075 p:0057 s:0407 e:000406 METHOD /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/simple.rb:93
c:0074 p:0007 s:0398 e:000397 METHOD /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/base.rb:68
c:0073 p:0009 s:0392 e:000391 BLOCK  /tmp/bundle/ruby/2.7.0/gems/i18n-1.8.2/lib/i18n/backend/fallbacks.rb:74 [
FINISH]
c:0072 p:---- s:0388 e:000387 CFUNC  :each
```

What makes me suspect they are related, is mostly the try to mark T_NONE object, and that they happen at random places, so my understanding is that it's inside the GC. But that's just speculation from myself.

#9 - 03/10/2020 08:20 PM - byroot (Jean Boussier)

So I've built a new ruby with that dumb patch:

```
diff --git a/vm_trace.c b/vm_trace.c
index 9a604814c6..dd5a44146f 100644
```

```
--- a/vm_trace.c
+++ b/vm_trace.c
@@ -1099,6 +1099,8 @@ tracepoint_attr_instruction_sequence(rb_execution_context_t *ec, VALUE tpval)
 static void
 tp_call_trace(VALUE tpval, rb_trace_arg_t *trace_arg)
 {
+    // HACK: TracePoint instance might have been GCed: https://bugs.ruby-lang.org/issues/16682
+    if (BUILTIN_TYPE(tpval) != T_DATA) return;
     rb_tp_t *tp = tpptr(tpval);

     if (tp->func) {
```

And no more crashes in sight, whereas with 2.7.0p0 way over 50% of the processes crash during the application boot. This makes me think they all have the same root cause.

#10 - 03/11/2020 01:05 AM - alanwu (Alan Wu)

It looks like 0c2d81da introduced this problem. On older versions the GC_EXIT event simply does not fire during shutdown.
The commit removed the usage of MATCH_ANY_FILTER_TH in the cleanup process which means GC events that used to not fire now do.
During filtering (see remove_event_hook()), it now only looks for hooks targeting the main thread while hooks made from alloc_event_hook() have their target thread set to null.
The commit message says it's a rename so I think this change was unintentional.
The following diff tries to restore the old behavior and seems to fix the problem:

```
diff --git a/eval.c b/eval.c
index f2fde81e19..08f7ba97de 100644
--- a/eval.c
+++ b/eval.c
@@ -26,6 +26,7 @@

 NORETURN(void rb_raise_jump(VALUE, VALUE));
 void rb_ec_clear_current_thread_trace_func(const rb_execution_context_t *ec);
+void rb_ec_clear_all_trace_func(const rb_execution_context_t *ec);

 static int rb_ec_cleanup(rb_execution_context_t *ec, volatile int ex);
 static int rb_ec_exec_node(rb_execution_context_t *ec, void *n);
@@ -140,7 +141,7 @@ rb_ec_teardown(rb_execution_context_t *ec)
     }
     EC_POP_TAG();
     rb_ec_exec_end_proc(ec);
-    rb_ec_clear_current_thread_trace_func(ec);
+    rb_ec_clear_all_trace_func(ec);
 }

 static void
diff --git a/vm_trace.c b/vm_trace.c
index 9a604814c6..241b929671 100644
--- a/vm_trace.c
+++ b/vm_trace.c
@@ -276,6 +276,12 @@ rb_ec_clear_current_thread_trace_func(const rb_execution_context_t *ec)
     rb_threadptr_remove_event_hook(ec, rb_ec_thread_ptr(ec), 0, Qundef);
 }

+void
+rb_ec_clear_all_trace_func(const rb_execution_context_t *ec)
+{
+    rb_threadptr_remove_event_hook(ec, MATCH_ANY_FILTER_TH, 0, Qundef);
+}
+
 /* invoke hooks */

 static void
```

I applied the diff on master and it avoids the crash on the original reproducer and my small 4-file reproducer

#11 - 03/11/2020 10:36 AM - byroot (Jean Boussier)

So I just tried your patch, it does indeed fix my reproduction script, however It doesn't fix the other crashes I reported above. However this time I managed to get the C-level backtrace for try to mark T_NONE object.

```
-- C level backtrace information -------------------------------------------
ruby-2.7.0/bin/ruby(rb_vm_bugreport+0x96) [0x10895e1f6]
ruby-2.7.0/bin/ruby(rb_bug+0xcc) [0x10896ab86]
ruby-2.7.0/bin/ruby(gc_mark_ptr+0x17a) [0x1087bc72a]
ruby-2.7.0/bin/ruby(mark_keyvalue+0x49) [0x1087bd4d9]
```

```
ruby-2.7.0/bin/ruby(st_general_foreach+0xa9) [0x1088c8389]
ruby-2.7.0/bin/ruby(rb_st_foreach+0x33) [0x1088c8a53]
ruby-2.7.0/bin/ruby(gc_mark_children+0x8e8) [0x1087b2078]
ruby-2.7.0/bin/ruby(gc_mark_stacked_objects_incremental+0x9e) [0x1087bae0e]
ruby-2.7.0/bin/ruby(newobj_slowpath+0x50f) [0x1087b7e9f]
ruby-2.7.0/bin/ruby(newobj_slowpath_wb_protected+0x14) [0x1087b7964]
ruby-2.7.0/bin/ruby(rb_hash_transform_keys+0x27) [0x1087c72e7]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(transform_values_foreach_replace+0x11) [0x1087cb181]
ruby-2.7.0/bin/ruby(st_general_foreach+0xe0) [0x1088c83c0]
ruby-2.7.0/bin/ruby(rb_hash_transform_values_bang+0x137) [0x1087c77e7]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(transform_values_foreach_replace+0x11) [0x1087cb181]
ruby-2.7.0/bin/ruby(st_general_foreach+0xe0) [0x1088c83c0]
ruby-2.7.0/bin/ruby(rb_hash_transform_values_bang+0x137) [0x1087c77e7]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(transform_values_foreach_replace+0x11) [0x1087cb181]
ruby-2.7.0/bin/ruby(st_general_foreach+0xe0) [0x1088c83c0]
ruby-2.7.0/bin/ruby(rb_hash_transform_values_bang+0x137) [0x1087c77e7]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(catch_i+0x67) [0x108959f17]
ruby-2.7.0/bin/ruby(vm_catch_protect+0xd5) [0x1089452a5]
ruby-2.7.0/bin/ruby(rb_f_catch+0x57) [0x108945987]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(rb_ary_each+0x39) [0x1086f5a69]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(catch_i+0x67) [0x108959f17]
ruby-2.7.0/bin/ruby(vm_catch_protect+0xd5) [0x1089452a5]
ruby-2.7.0/bin/ruby(rb_f_catch+0x57) [0x108945987]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_funcallv_with_cc+0x8a) [0x10893d73a]
ruby-2.7.0/bin/ruby(rb_inspect+0x20) [0x10882c7c0]
ruby-2.7.0/bin/ruby(inspect_ary+0x98) [0x1087050b8]
ruby-2.7.0/bin/ruby(exec_recursive+0x423) [0x108900943]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x38df) [0x1089350af]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_call0+0x5e7) [0x108958ec7]
ruby-2.7.0/bin/ruby(rb_funcall_with_block_kw+0x85) [0x108942875]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(rb_ary_collect+0xf2) [0x1086fbdd2]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(rb_ary_each+0x39) [0x1086f5a69]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield_values2+0x4e) [0x108942cde]
ruby-2.7.0/bin/ruby(rb_ensure+0xf0) [0x10879c430]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
```

```
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_call_opt_send+0x2f4) [0x108950074]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(rb_ary_each+0x39) [0x1086f5a69]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_call_opt_send+0x2f4) [0x108950074]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_call_opt_send+0x2f4) [0x108950074]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(yield_under+0x40d) [0x108944f4d]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_call_opt_send+0x2f4) [0x108950074]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_vm_invoke_bmethod+0x7f1) [0x108947821]
ruby-2.7.0/bin/ruby(vm_call_bmethod+0xac) [0x10894fbfc]
ruby-2.7.0/bin/ruby(vm_call_opt_send+0x2f4) [0x108950074]
ruby-2.7.0/bin/ruby(vm_exec_core+0x38df) [0x1089350af]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(rb_ary_each+0x39) [0x1086f5a69]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_yield+0xa7) [0x108942a87]
ruby-2.7.0/bin/ruby(rb_ary_collect+0xf2) [0x1086fbdd2]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_proc_call+0x9f) [0x10886255f]
ruby-2.7.0/bin/ruby(rb_ec_exec_end_proc+0x172) [0x10879e4c2]
ruby-2.7.0/bin/ruby(rb_ec_teardown+0xaf) [0x10879afbf]
ruby-2.7.0/bin/ruby(rb_ec_cleanup+0x17e) [0x10879b18e]
ruby-2.7.0/bin/ruby(ruby_stop+0x9) [0x10879b3d9]
ruby-2.7.0/bin/ruby(rb_f_fork+0x98) [0x10886e028]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x492b) [0x1089360fb]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(loop_i+0x29) [0x108959f99]
ruby-2.7.0/bin/ruby(rb_vrescue2+0x114) [0x10879c024]
ruby-2.7.0/bin/ruby(rb_rescue2+0x7b) [0x10879beeb]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x3782) [0x108934f52]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(load_iseq_eval+0xb9) [0x1087f4fb9]
ruby-2.7.0/bin/ruby(require_internal+0x2f1) [0x1087f3cc1]
ruby-2.7.0/bin/ruby(rb_f_require+0x21) [0x1087f3201]
ruby-2.7.0/bin/ruby(vm_call_cfunc+0x170) [0x10894f220]
ruby-2.7.0/bin/ruby(vm_exec_core+0x38df) [0x1089350af]
ruby-2.7.0/bin/ruby(rb_vm_exec+0xadc) [0x10894a03c]
ruby-2.7.0/bin/ruby(rb_ec_exec_node+0xc6) [0x10879b5a6]
```

```
ruby-2.7.0/bin/ruby(ruby_run_node+0x55) [0x10879b485]
ruby-2.7.0/bin/ruby(main+0x5d) [0x1086f2c9d]
```

#12 - 03/11/2020 04:16 PM - byroot (Jean Boussier)

I now think the TracePoint issue is a red herring, I figured a way to reproduce what I think is the underlying root cause without any TracePoint enabled: https://bugs.ruby-lang.org/issues/16689

We can probably close this ticket.

#13 - 03/11/2020 04:27 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

#14 - 03/16/2020 04:36 AM - mame (Yusuke Endoh)

- Status changed from Closed to Assigned

- Assignee set to ko1 (Koichi Sasada)

#15 - 03/16/2020 04:44 AM - naruse (Yui NARUSE)

- Backport changed from 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED

#16 - 03/16/2020 04:37 PM - alanwu (Alan Wu)

Posting here in case it gets lost. This issue is unrelated to #16689.
I have a patch for this issue at https://github.com/ruby/ruby/pull/2959.

#17 - 03/30/2020 03:41 AM - alanwu (Alan Wu)

- Status changed from Assigned to Closed

Applied in changeset git|b385f7670ffa420790bc548747fa4b58c4c5d8f6.

Clear all trace events during teardown

Since 0c2d81dada, not all trace events are cleared during VM teardown.
This causes a crash when there is a tracepoint for
RUBY_INTERNAL_EVENT_GC_EXIT active during teardown.

The commit looks like a refactoring commit so I think this change was
unintentional.

[Bug #16682]

#18 - 03/30/2020 11:30 AM - naruse (Yui NARUSE)

- Backport changed from 2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONE

ruby_2_7 b5fa156b7907b8ea1baf8b9f0cb6a66b0fecb3d4.