Ruby - Bug #17573

Crashes in profiling tools when signals arrive in non-Ruby threads

01/23/2021 12:08 AM - jhawthorn (John Hawthorn)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x86_64-darwin19]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: DONE

Description

Stackprof (and likely similar tools) works by setting up a timer to sends it a unix signal on an interval. From that signal handler it does a small amount of internal bookkeeping and calls rb_postponed_job_register_one.

This is a problem because unix signals arrive on an arbitrary thread, and as of Ruby 3.0 the execution context (which rb_postponed_job_register_one relies on) is stored as a thread-local.

This reproduction crashes reliably for me on macos. It doesn't seem to on linux, maybe because the timer thread is different or the kernel has a different "arbitrary" choice. It feels like this is just one of the circumstances this crash could happen.

```
require "stackprof"

StackProf.run(interval: 100) do
  1000.times do
    GC.start
  end
end
```

```
$ ruby crash_stackprof.rb
[BUG] Segmentation fault at 0x0000000000000038
ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [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.

-- Machine register context ------------------------------------------------
 rax: 0x0000000000000000 rbx: 0x0000000107fbb780 rcx: 0x0000000000000000
 rdx: 0x0000000000000000 rdi: 0x0000000106982c28 rsi: 0x0000000107fbb780
 rbp: 0x000070000eb47a10 rsp: 0x000070000eb479f0  r8: 0x000070000eb47eb0
  r9: 0xd44931e7344c235f r10: 0x00007fff6ef49501 r11: 0x0000000000000202
 r12: 0xd44931e7344c235f r13: 0x00000000ffffffff r14: 0x0000000000000000
 r15: 0x0000000000000000 rip: 0x00000001068c85fd rfl: 0x0000000000010202

-- C level backtrace information -------------------------------------------
/Users/jhawthorn/.rubies/ruby-3.0.0/bin/ruby(rb_vm_bugreport+0x6cf) [0x1068c2d5f]
/Users/jhawthorn/.rubies/ruby-3.0.0/bin/ruby(rb_bug_for_fatal_signal+0x1d6) [0x1066dc556]
/Users/jhawthorn/.rubies/ruby-3.0.0/bin/ruby(sigsegv+0x5b) [0x10681aa0b]
/usr/lib/system/libsystem_platform.dylib(_sigtramp+0x1d) [0x7fff6efff5fd]
/Users/jhawthorn/.rubies/ruby-3.0.0/bin/ruby(rb_postponed_job_register_one+0x1d) [0x1068c85fd]
/usr/lib/system/libsystem_platform.dylib(0x7fff6efff5fd) [0x7fff6efff5fd]
```

0x38 is the address of ((rb_execution_context_t *)0)->vm.

lldb shows that it comes from a second thread which was running timer_pthread_fn

```
$ lldb =ruby -- ./crash_stackprof.rb
(lldb) target create "/Users/jhawthorn/.rubies/ruby-3.0.0/bin/ruby"
```

```
                                    ruCurrent executable set to '/Users/jhawthorn/.rubies/ruby-3.0
.0/bin/ruby' (x86_64).
(lldb) settings set -- target.run-args  "./crash_stackprof.rb"

                                    (lldb) run
Process 92893 launched: '/Users/jhawthorn/.rubies/ruby-3.0.0/bin/ruby' (x86_64)

                                    Process 92893 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = signal SIGALRM
    frame #0: 0x00000001000dfbcd ruby`rgengc_check_relation [inlined] RVALUE_OLD_P_RAW(obj=4303689
480) at gc.c:1419:32
   1416 RVALUE_OLD_P_RAW(VALUE obj)
   1417 {
   1418     const VALUE promoted = FL_PROMOTED0 | FL_PROMOTED1;
-> 1419     return (RBASIC(obj)->flags & promoted) == promoted;
   1420 }
   1421
   1422 static inline int

                                      thread #2, stop reason = EXC_BAD_ACCESS (code=1, address=0x3
8)
    frame #0: 0x0000000010029a5fd ruby`rb_postponed_job_register_one(flags=3492904, func=(stackprof
.bundle`stackprof_gc_job_handler at stackprof.c:598), data=0x0000000000000000) at vm_trace.c:1622:
19                                    1619 rb_postponed_job_register_one(unsigned int flags, rb_p
ostponed_job_func_t func, void *data)
   1620 {

                                      1621     rb_execution_context_t *ec = GET_EC();
-> 1622     rb_vm_t *vm = rb_ec_vm_ptr(ec);
   1623     rb_postponed_job_t *pjob;
   1624     rb_atomic_t i, index;
   1625
Target 0: (ruby) stopped.
(lldb) t 2
* thread #2
    frame #0: 0x0000000010029a5fd ruby`rb_postponed_job_register_one(flags=3492904, func=(stackprof
.bundle`stackprof_gc_job_handler at stackprof.c:598), data=0x0000000000000000) at vm_trace.c:1622:
19
   1619 rb_postponed_job_register_one(unsigned int flags, rb_postponed_job_func_t func, void *data
)
   1620 {
   1621     rb_execution_context_t *ec = GET_EC();
-> 1622     rb_vm_t *vm = rb_ec_vm_ptr(ec);
   1623     rb_postponed_job_t *pjob;
   1624     rb_atomic_t i, index;
   1625
(lldb) bt
* thread #2
  * frame #0: 0x0000000010029a5fd ruby`rb_postponed_job_register_one(flags=3492904, func=(stackprof
.bundle`stackprof_gc_job_handler at stackprof.c:598), data=0x0000000000000000) at vm_trace.c:1622:
19
    frame #1: 0x00007fff6efff5fd libsystem_platform.dylib`_sigtramp + 29
    frame #2: 0x00007fff6ef4e3d7 libsystem_kernel.dylib`poll + 11
    frame #3: 0x0000000100238e1e ruby`timer_pthread_fn(p=<unavailable>) at thread_pthread.c:2189:1
5
    frame #4: 0x00007fff6f00b109 libsystem_pthread.dylib`_pthread_start + 148
    frame #5: 0x00007fff6f006b8b libsystem_pthread.dylib`thread_start + 15
```

Attached is my attempted fix (also available at https://github.com/ruby/ruby/pull/4108) which uses the main-ractor's EC if there is none on the current thread. I *hope* this works (it seems to and fixes the crash) because before Ruby 3.0 there was a global EC, but I'm not entirely sure if this will cause other problems.

If accepted this should be backported to the 3.0 branch.

Related issues:

Related to Ruby - Bug #15263: [PATCH] vm_trace.c (postponed_job_register): on...	**Assigned**

Associated revisions

Revision f5d20411386ff2552ff27661387ddc4bae1ebc30 - 11/23/2021 12:29 AM - alanwu (Alan Wu)

Avoid assert failure when NULL EC is expected

After 5680c38c75aeb5cbd219aafa8eb48c315f287d97, postponed job APIs now
expect to be called on native threads not managed by Ruby and handles
getting a NULL execution context. However, in debug builds the change
runs into an assertion failure with GET_EC() which asserts that EC is
non-NULL. Avoid the assertion failure by passing false for expect_ec
instead as the intention is to handle when there is no EC.

Add a test from John Crepezzi and John Hawthorn to exercise this
situation.

See GH-4108
See GH-5094

[Bug #17573]

Co-authored-by: John Hawthorn john@hawthorn.email
Co-authored-by: John Crepezzi john.crepezzi@gmail.com

Revision f5d20411386ff2552ff27661387ddc4bae1ebc30 - 11/23/2021 12:29 AM - alanwu (Alan Wu)

Avoid assert failure when NULL EC is expected

After 5680c38c75aeb5cbd219aafa8eb48c315f287d97, postponed job APIs now
expect to be called on native threads not managed by Ruby and handles
getting a NULL execution context. However, in debug builds the change
runs into an assertion failure with GET_EC() which asserts that EC is
non-NULL. Avoid the assertion failure by passing false for expect_ec
instead as the intention is to handle when there is no EC.

Add a test from John Crepezzi and John Hawthorn to exercise this
situation.

See GH-4108
See GH-5094

[Bug #17573]

Co-authored-by: John Hawthorn john@hawthorn.email
Co-authored-by: John Crepezzi john.crepezzi@gmail.com

Revision f5d20411 - 11/23/2021 12:29 AM - alanwu (Alan Wu)

Avoid assert failure when NULL EC is expected

After 5680c38c75aeb5cbd219aafa8eb48c315f287d97, postponed job APIs now
expect to be called on native threads not managed by Ruby and handles
getting a NULL execution context. However, in debug builds the change
runs into an assertion failure with GET_EC() which asserts that EC is
non-NULL. Avoid the assertion failure by passing false for expect_ec
instead as the intention is to handle when there is no EC.

Add a test from John Crepezzi and John Hawthorn to exercise this
situation.

See GH-4108
See GH-5094

[Bug #17573]

Co-authored-by: John Hawthorn john@hawthorn.email
Co-authored-by: John Crepezzi john.crepezzi@gmail.com

Revision 949af69408e44b69cc7437b58e8edbe3cd77c966 - 11/23/2021 05:52 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 5680c38c75aeb5cbd219aafa8eb48c315f287d97,f5d20411386ff2552ff27661387ddc4bae1ebc30: [Backport #17573]

```
    Use valid `ec` for postponed job.

    Postponed job can be registered from non-Ruby thread, which means
    `ec` in TLS can be NULL. In this case, use main thread's `ec` instead.
```

```
    See https://github.com/ruby/ruby/pull/4108
    and https://github.com/ruby/ruby/pull/4336
    ---
     vm_trace.c | 16 ++++++++++++----
     1 file changed, 12 insertions(+), 4 deletions(-)

    Avoid assert failure when NULL EC is expected

    After 5680c38c75aeb5cbd219aafa8eb48c315f287d97, postponed job APIs now
    expect to be called on native threads not managed by Ruby and handles
    getting a NULL execution context. However, in debug builds the change
    runs into an assertion failure with GET_EC() which asserts that EC is
    non-NULL. Avoid the assertion failure by passing `false` for `expect_ec`
    instead as the intention is to handle when there is no EC.

    Add a test from John Crepezzi and John Hawthorn to exercise this
    situation.

    See GH-4108
    See GH-5094

    [Bug #17573]

    Co-authored-by: John Hawthorn <john@hawthorn.email>
    Co-authored-by: John Crepezzi <john.crepezzi@gmail.com>
    ---
     ext/-test-/postponed_job/postponed_job.c       | 31 ++++++++++++++++++++++++++++
     test/-ext-/postponed_job/test_postponed_job.rb |  7 ++++++
     vm_trace.c                                     |  2 +-
     3 files changed, 39 insertions(+), 1 deletion(-)
```

Revision 949af69408e44b69cc7437b58e8edbe3cd77c966 - 11/23/2021 05:52 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 5680c38c75aeb5cbd219aafa8eb48c315f287d97,f5d20411386ff2552ff27661387ddc4bae1ebc30: [Backport #17573]

```
    Use valid `ec` for postponed job.

    Postponed job can be registered from non-Ruby thread, which means
    `ec` in TLS can be NULL. In this case, use main thread's `ec` instead.

    See https://github.com/ruby/ruby/pull/4108
    and https://github.com/ruby/ruby/pull/4336
    ---
     vm_trace.c | 16 ++++++++++++----
     1 file changed, 12 insertions(+), 4 deletions(-)

    Avoid assert failure when NULL EC is expected

    After 5680c38c75aeb5cbd219aafa8eb48c315f287d97, postponed job APIs now
    expect to be called on native threads not managed by Ruby and handles
    getting a NULL execution context. However, in debug builds the change
    runs into an assertion failure with GET_EC() which asserts that EC is
    non-NULL. Avoid the assertion failure by passing `false` for `expect_ec`
    instead as the intention is to handle when there is no EC.

    Add a test from John Crepezzi and John Hawthorn to exercise this
    situation.

    See GH-4108
    See GH-5094

    [Bug #17573]

    Co-authored-by: John Hawthorn <john@hawthorn.email>
    Co-authored-by: John Crepezzi <john.crepezzi@gmail.com>
    ---
     ext/-test-/postponed_job/postponed_job.c       | 31 ++++++++++++++++++++++++++++
     test/-ext-/postponed_job/test_postponed_job.rb |  7 ++++++
     vm_trace.c                                     |  2 +-
     3 files changed, 39 insertions(+), 1 deletion(-)
```

Revision 949af694 - 11/23/2021 05:52 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 5680c38c75aeb5cbd219aafa8eb48c315f287d97,f5d20411386ff2552ff27661387ddc4bae1ebc30: [Backport #17573]

```
    Use valid `ec` for postponed job.

    Postponed job can be registered from non-Ruby thread, which means
    `ec` in TLS can be NULL. In this case, use main thread's `ec` instead.

    See https://github.com/ruby/ruby/pull/4108
    and https://github.com/ruby/ruby/pull/4336
    ---
     vm_trace.c | 16 ++++++++++++----
     1 file changed, 12 insertions(+), 4 deletions(-)

    Avoid assert failure when NULL EC is expected

    After 5680c38c75aeb5cbd219aafa8eb48c315f287d97, postponed job APIs now
    expect to be called on native threads not managed by Ruby and handles
    getting a NULL execution context. However, in debug builds the change
    runs into an assertion failure with GET_EC() which asserts that EC is
    non-NULL. Avoid the assertion failure by passing `false` for `expect_ec`
    instead as the intention is to handle when there is no EC.

    Add a test from John Crepezzi and John Hawthorn to exercise this
    situation.

    See GH-4108
    See GH-5094

    [Bug #17573]

    Co-authored-by: John Hawthorn <john@hawthorn.email>
    Co-authored-by: John Crepezzi <john.crepezzi@gmail.com>
    ---
     ext/-test-/postponed_job/postponed_job.c       | 31 +++++++++++++++++++++++++++
     test/-ext-/postponed_job/test_postponed_job.rb |  7 ++++++
     vm_trace.c                                     |  2 +-
     3 files changed, 39 insertions(+), 1 deletion(-)
```

History

#1 - 01/24/2021 11:11 PM - jhawthorn (John Hawthorn)

- File use_main_ractor_ec_on_threads_without_ec.patch added

#2 - 01/29/2021 09:20 AM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)

#3 - 02/18/2021 07:58 AM - ko1 (Koichi Sasada)

Ah, OK. This issue doesn't expose on recent Linux system.
Hmm. How to debug it...

#4 - 02/18/2021 02:08 PM - byroot (Jean Boussier)

> This issue doesn't expose on recent Linux system

I do have 3.0 receive SEGV avout 50% of the time when using stackprof in CPU mode on ubuntu 20.04. Unfortunately All I get as output is Segmentation fault, so I don't know wether it's the same issue or not.

#5 - 03/03/2021 03:57 PM - byroot (Jean Boussier)

> I don't know wether it's the same issue or not.

So I tested this patch on top of the current ruby_3_0 branch, and it does fix the stackprof issue I had.

#6 - 03/30/2021 12:15 AM - alanwu (Alan Wu)

> Ah, OK. This issue doesn't expose on recent Linux system.

I can somewhat reliably repro by running ruby --jit repro.rb with the following:

```
# repro.rb

require "stackprof"

StackProf.run(interval: 100) do
  1000.times do
    GC.start
  end
end
```

I get a crash with the ruby:3.0.0 image from DockerHub.

It happens with --jit because MJIT spawns a worker thread that doesn't have
an execution context. It could happen with the timer thread too, but not all build
configurations spawn the timer thread.

Since this problem is specific to rb_postponed_job_register_one(), I think it
would be better to fix it there rather than touching the much more popular rb_current_execution_context():

```
diff --git a/vm_trace.c b/vm_trace.c
index 383f255799..b012e946e9 100644
--- a/vm_trace.c
+++ b/vm_trace.c
@@ -1599,8 +1599,8 @@ postponed_job_register(rb_execution_context_t *ec, rb_vm_t *vm,
 int
 rb_postponed_job_register(unsigned int flags, rb_postponed_job_func_t func, void *data)
 {
-    rb_execution_context_t *ec = GET_EC();
-    rb_vm_t *vm = rb_ec_vm_ptr(ec);
+    rb_vm_t *vm = GET_VM();
+    rb_execution_context_t *ec = rb_vm_main_ractor_ec(vm);

   begin:
     switch (postponed_job_register(ec, vm, flags, func, data, MAX_POSTPONED_JOB, vm->postponed_job_index)) {
@@ -1618,8 +1618,8 @@ rb_postponed_job_register(unsigned int flags, rb_postponed_job_func_t func, void
 int
 rb_postponed_job_register_one(unsigned int flags, rb_postponed_job_func_t func, void *data)
 {
-    rb_execution_context_t *ec = GET_EC();
-    rb_vm_t *vm = rb_ec_vm_ptr(ec);
+    rb_vm_t *vm = GET_VM();
+    rb_execution_context_t *ec = rb_vm_main_ractor_ec(vm);
     rb_postponed_job_t *pjob;
     rb_atomic_t i, index;
```

Basically make the postpond job API always deliver to the main ractor. I think it makes sense
given that signal don't necessarily land on a Ruby thread.

Available as a GitHub PR: https://github.com/ruby/ruby/pull/4336

#7 - 07/07/2021 06:09 PM - alanwu (Alan Wu)

- Related to Bug #15263: [PATCH] vm_trace.c (postponed_job_register): only hit main thread added

#8 - 11/22/2021 10:49 PM - alanwu (Alan Wu)

- Backport changed from 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN to 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: REQUIRED

#9 - 11/23/2021 01:05 AM - alanwu (Alan Wu)

- Status changed from Open to Closed

Applied in changeset git|f5d20411386ff2552ff27661387ddc4bae1ebc30.

Avoid assert failure when NULL EC is expected

After 5680c38c75aeb5cbd219aafa8eb48c315f287d97, postponed job APIs now
expect to be called on native threads not managed by Ruby and handles
getting a NULL execution context. However, in debug builds the change

runs into an assertion failure with GET_EC() which asserts that EC is non-NULL. Avoid the assertion failure by passing false for expect_ec instead as the intention is to handle when there is no EC.

Add a test from John Crepezzi and John Hawthorn to exercise this situation.

See GH-4108
See GH-5094

[Bug #17573]

Co-authored-by: John Hawthorn john@hawthorn.email
Co-authored-by: John Crepezzi john.crepezzi@gmail.com

#10 - 11/23/2021 06:13 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: REQUIRED to 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: DONE

ruby_3_0 949af69408e44b69cc7437b58e8edbe3cd77c966 merged revision(s) 5680c38c75aeb5cbd219aafa8eb48c315f287d97,f5d20411386ff2552ff27661387ddc4bae1ebc30.

Files

use_main_ractor_ec_on_threads_without_ec.patch	3.28 KB	01/23/2021	jhawthorn (John Hawthorn)
use_main_ractor_ec_on_threads_without_ec.patch	3.84 KB	01/24/2021	jhawthorn (John Hawthorn)