Ruby master - Bug #14725
fatal: No live threads left. Deadlock? since Revision 63215
04/30/2018 10:31 PM - yahonda (Yasuo Honda)

| Status: | Closed |
| Priority: | Normal |
| Assignee: | |
| Target version: | ruby 2.6.0dev (2018-04-30 trunk 63306) [x86_64-linux] |
| Backport: | 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: REQUIRED |

**Description**

**Steps to reproduce**

```
$ sudo gem install bundler
$ git clone https://github.com/rails/rails.git
$ cd rails/railties
$ bundle install
$ bundle exec ruby -w -Itest test/application/asset_debugging_test.rb -n test_assets_are_concatenated_when_debug_is_off_and_compile_is_off_either_if_debug_assets_param_is_provided
```

**Expected behavior**

It should pass as Ruby 2.5.1.

**Actual result**

It always gets an error as below:

```
$ bundle exec ruby -w -Itest test/application/asset_debugging_test.rb -n test_assets_are_concatenated_when_debug_is_off_and_compile_is_off_either_if_debug_assets_param_is_provided
Run options: -n test_assets_are_concatenated_when_debug_is_off_and_compile_is_off_either_if_debug_assets_param_is_provided --seed 35038

# Running:

e
Error: ApplicationTests::AssetDebuggingTest#test_assets_are_concatenated_when_debug_is_off_and_compile_is_off_either_if_debug_assets_param_is_provided: 
RuntimeError: rails command failed (1): bin/rails assets:precompile --trace 2>&1
  ** Invoke assets:precompile (first_time)
  ** Invoke assets:environment (first_time)
  ** Execute assets:environment
  ** Invoke yarn:install (first_time)
  ** Execute yarn:install
Yarn executable was not detected in the system.
Download Yarn at https://yarnpkg.com/en/docs/install
```

fatal: No live threads left. Deadlock?
5 threads, 5 sleeps current: 0x000055f8f8565d50 main thread: 0x000055f8f8565d50
* #<Thread:0x000055f8f8087f408/u...
This issue has been opened to Rails https://github.com/rails/rails/issues/32760
According to git bisect r63215 https://svn.ruby-lang.org/cgi-bin/viewvc.cgi?revision=63215&view=revision triggers this error.

Associated revisions

Revision d2f52a5f - 04/30/2018 11:47 PM - normal

thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[Bug #14725] [Bug #14634]
Revision 63309 - 04/30/2018 11:47 PM - normalperson (Eric Wong)
thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[Bug #14725] [Bug #14634]

Revision 63309 - 04/30/2018 11:47 PM - normal
thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[Bug #14725] [Bug #14634]

Revision f5db4a0c - 04/30/2018 11:53 PM - normal

thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[Bug #14725] [Bug #14634]

Revision 63310 - 04/30/2018 11:53 PM - normalperson (Eric Wong)
test/thread/test_cv.rb: test CV usability inside forked child

- test/thread/test_cv.rb (def test_condvarfork): new test [Bug #14725]

Revision 63310 - 04/30/2018 11:53 PM - normal

test/thread/test_cv.rb: test CV usability inside forked child

- test/thread/test_cv.rb (def test_condvarfork): new test [Bug #14725]

Revision 63d9ab33 - 01/23/2019 02:14 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 62934,63210,63215,63309: [Backport #14634]
thread_sync.c: avoid reaching across stacks of dead threads

rb_ensure is insufficient cleanup for fork and we must reinitialize all waitqueues in the child process.

Unfortunately this increases the footprint of ConditionVariable, Queue and SizedQueue by 8 bytes on 32-bit (16 bytes on 64-bit).

[ruby-core:86316] [Bug #14634]

variable.c: fix thread + fork errors in autoload

This is fairly non-intrusive bugfix to prevent children from trying to reach into thread stacks of the parent. I will probably reuse this idea and redo r62934, too (same bug).

* vm_core.h (typedef struct rb_vm_struct): add fork_gen counter
* thread.c (rb_thread_atfork_internal): increment fork_gen
* variable.c (struct autoload_data_i): store fork_gen
* variable.c (check_autoload_data): remove (replaced with get_...)
* variable.c (get_autoload_data): check fork_gen when retrieving
* variable.c (check_autoload_required): use get_autoload_data
* variable.c (rb_autoloading_value): ditto
* variable.c (rb_autoload_p): ditto
* variable.c (current_autoload_data): ditto
* variable.c (autoload_reset): reset fork_gen, adjust indent
* variable.c (rb_autoload_load): set fork_gen when setting state
* test/ruby/test_autoload.rb (test_autoload_fork): new test
[ruby-core:86410] [Bug #14634]

thread_sync: redo r62934 to use fork_gen

Instead of maintaining linked-lists to store all
rb_queue/rb_szqueue/rb_convard struct; store only a fork_gen serial number to simplify management of these items.

This reduces initialization costs and avoids the up-front cost of resetting all Queue/SizedQueue/ConditionVariable objects at fork while saving 8 bytes per-structure on 64-bit. There are no savings on 32-bit.

* thread.c (rb_thread_atfork_internal): remove rb_thread_sync_reset_all call
* thread_sync.c (rb_thread_sync_reset_all): remove
* thread_sync.c (queue_live): remove
* thread_sync.c (queue_free): remove
* thread_sync.c (struct rb_queue): s/live/fork_gen/
* thread_sync.c (queue_data_type): use default free
* thread_sync.c (queue_alloc): remove list_add
* thread_sync.c (queue_fork_check): new function
* thread_sync.c (queue_ptr): call queue_fork_check
* thread_sync.c (szqueue_free): remove
* thread_sync.c (szqueue_data_type): use default free
* thread_sync.c (szqueue_alloc): remove list_add
* thread_sync.c (szqueue_ptr): check fork_gen via queue_fork_check
* thread_sync.c (struct rb_condvar): s/live/fork_gen/
* thread_sync.c (condvar_free): remove
* thread_sync.c (cv_data_type): use default free
* thread_sync.c (condvar_ptr): check fork_gen
* thread_sync.c (condvar_alloc): remove list_add
* thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong.

[Bug #14725] [Bug #14634]

Revision 66912 - 01/23/2019 02:14 PM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 62934,63210,63215,63309: [Backport #14634]
thread_sync.c: avoid reaching across stacks of dead threads
rb_ensure is insufficient cleanup for fork and we must reinitialize all waitqueues in the child process.
Unfortunately this increases the footprint of ConditionVariable, Queue and SizedQueue by 8 bytes on 32-bit (16 bytes on 64-bit).

[Bug #14634]
variable.c: fix thread + fork errors in autoload
This is fairly non-intrusive bugfix to prevent children from trying to reach into thread stacks of the parent. I will probably reuse this idea and redo r62934, too (same bug).

* vm_core.h (typedef struct rb_vm_struct): add fork_gen counter
* thread.c (rb_thread_atfork_internal): increment fork_gen
* variable.c (struct autoload_data_i): store fork_gen
* variable.c (check autoload data): remove (replaced with get...)
* variable.c (get autoload data): check fork_gen when retrieving
* variable.c (check autoload required): use get autoload data
* variable.c (rb autoload loading value): ditto
* variable.c (rb autoload p): ditto
* variable.c (currentautoload data): ditto
* variable.c (autoload reset): reset fork_gen, adjust indent
* variable.c (rb autoload load): set fork_gen when setting state
* test/ruby/test autoload.rb (test autoload fork): new test

[Bug #14634]

thread_sync: redo r62934 to use fork_gen
Instead of maintaining linked-lists to store all
rb_queue/rb_szqueue/rb_condvar structs; store only a fork_gen serial number to simplify management of these items.

This reduces initialization costs and avoids the up-front cost of resetting all Queue/SizedQueue/ConditionVariable objects at fork while saving 8 bytes per-structure on 64-bit. There are no savings on 32-bit.

* thread.c (rb_thread_atfork_internal): remove rb_thread_sync_reset_all call
* thread_sync.c (rb_thread_sync_reset_all): remove
* thread_sync.c (queue_live): remove
* thread_sync.c (queue_free): remove
* thread_sync.c (struct rb_queue): s/live/fork_gen/
* thread_sync.c (queue_data_type): use default free
* thread_sync.c (queue_alloc): remove list_add
* thread_sync.c (queue_fork_check): new function
* thread_sync.c (queue_ptr): call queue_fork_check
* thread_sync.c (szqueue_free): remove
* thread_sync.c (szqueue_data_type): use default free
* thread_sync.c (szqueue_alloc): remove list_add
* thread_sync.c (szqueue_ptr): check fork_gen via queue_fork_check
* thread_sync.c (struct rb_condvar): s/live/fork_gen/
* thread_sync.c (condvar_free): remove
* thread_sync.c (cv_data_type): use default free
* thread_sync.c (condvar_ptr): check fork_gen
* thread_sync.c (condvar_alloc): remove list_add

UPDATE

thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[BUG #14725] [BUG #14634]

Revision 703d961 - 03/14/2019 10:21 PM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 63309:

thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[BUG #14725] [BUG #14634]

Revision 67259 - 03/14/2019 10:21 PM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 63309:

thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[BUG #14725] [BUG #14634]

History

#1 - 04/30/2018 11:33 PM - normalperson (Eric Wong)
Investigating...

#2 - 04/30/2018 11:47 PM - normalperson (Eric Wong)
- Status changed from Open to Closed

Applied in changeset trunk/r63309.

thread_sync.c (condvar_ptr): reset fork_gen after forking

Otherwise the condition variable waiter list will always be empty, which is wrong :x

[BUG #14725] [BUG #14634]
be empty, which is wrong :x

[Bug #14725] [Bug #14634]

#3 - 05/01/2018 12:05 AM - normalperson (Eric Wong)
yasu.honda@gmail.com wrote:

fatal: No live threads left. Deadlock?
5 threads, 5 sleeps current:0x00005f8f8565d50 main thread:0x00005f8f8565d50

- # rb_thread_t:0x00005f8f8565d50 native:0x00007c6ea4bf700 int:0
  /usr/local/lib/ruby/gems/2.6.0/gems/concurrent-ruby-1.0.5/lib/concurrent/synchronization/mri_lockable_object.rb:43:in sleep'
  /usr/local/lib/ruby/gems/2.6.0/gems/concurrent-ruby-1.0.5/lib/concurrent/synchronization/mri_lockable_object.rb:43:in:wait'
  /usr/local/lib/ruby/gems/2.6.0/gems/concurrent-ruby-1.0.5/lib/concurrent/synchronization/mri_lockable_object.rb:43:in:ns_wait'
  /usr/local/lib/ruby/gems/2.6.0/gems/concurrent-ruby-1.0.5/lib/concurrent/synchronization/abstract_lockable_object.rb:43:in:ns_wait_until'
  /usr/local/lib/ruby/gems/2.6.0/gems/concurrent-ruby-1.0.5/lib/concurrent/atomic/event.rb:87:in:block in wait'
  /usr/local/lib/ruby/gems/2.6.0/gems/concurrent-ruby-1.0.5/lib/concurrent/atomic/event.rb:84:in:wait'
  /usr/local/lib/ruby/gems/2.6.0/gems/concurrent-ruby-1.0.5/lib/concurrent/concern/obligation.rb:75:in:wait'

It looks like condvar fix for r63309 (and r63310 for test) will work.
(Sorry, didn't test with rails, ENOSPC)

#4 - 05/01/2018 10:35 AM - yahonda (Yasuo Honda)

Thanks for the update. I have confirmed this error does not reproduce with revision 63314.

```
$ ruby -v
ruby 2.6.0dev (2018-05-01 trunk 63314) {x86_64-linux}
$ bundle exec ruby -w -Itest/test/application/asset_debugging_test.rb -n test_assets_are_concatenated_when_debug_is_off_and_compile_is_off_either_if_debug_assets_param_is_provided
Run options: -n test_assets_are_concatenated_when_debug_is_off_and_compile_is_off_either_if_debug_assets_param_is_provided --seed 19317

# Running:
.
```

Finished in 8.251301s, 0.1212 runs/s, 0.4848 assertions/s.
1 runs, 4 assertions, 0 failures, 0 errors, 0 skips

#5 - 03/14/2019 02:40 PM - k0kubun (Takashi Kokubun)

- Backport changed from 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN to 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: REQUIRED

r63215 is backported to ruby_2_5 in r66912 but its bug fix r63309 is not backported to ruby_2_5 yet. I confirmed that
https://github.com/sorah/repro254stuck sticks with ruby_2_5 branch (r67242) and it's fixed by cherry-picking r63309.

nagachika (Tomoyuki Chikanaga) Could you backport r63309 to ruby_2_5 branch?