Ruby - Bug #14634

Queue#push seems to crash after fork

03/26/2018 11:16 PM - zetaben (Benoit Larroque)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.5.1p57 (2018-03-29 revision 63029) [x86_64-linux]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: DONE

Description

Hello,

We are providing an agent that uses dynamic instrumentation to provide additional security checks on Rails applications.

We recently received issues from one of our customers that just upgraded from Ruby 2.4 to 2.5.0 on Heroku about segfault coming from inside the gem. These seemed to happen randomly when our agent was pushing to a Ruby Queue (that is consumed by our reporting thread).

We managed to reproduced this issue locally using the customer application code. It seems to happen regularly to the first Queue#push done by the instrumented Unicorn worker just after startup:

- Master start-up
- Our agent starts, connects to our backend, apply instrumentation, creates the queue and starts listening on it (Queue#pop interrupted by a Timeout in an infinite loop)
- Master fork and launch worker
- Worker starts to answer HTTP traffic
- Agent detects that reporting thread is dead (because of fork) and restarts a thread listening on the same Queue instance but in the new process
- Worker handles another HTTP request and tries to Queue#push some metrics to Queue instance

==> Ruby then crashes with either:

- Most often: [BUG] pthread_mutex_lock: Invalid argument (EINVAL)
- Sometime with [BUG] Segmentation fault at 0x0000000000000038

Both being located to:

```
c:0052 p:---- s:0384 e:000383 CFUNC  :push
```

which would be the C-func that pushes in the Queue.

I'm attaching the stripped bug report traces to ticket.

We found a workaround by actually recreating a new Queue instance on agent start in the worker (So it does not happen on a brand new Queue after forking).

Also, the first workaround we tried though was copying the pure Ruby Queue implementation from previous Ruby versions (circa 2.0.0). This actually didn't work and lead to a similar crash happening this time on the ConditionVariable

```
2018-03-21T22:53:41.973451+00:00 app[web.2]: /app/vendor/bundle/ruby/2.5.0/gems/sqreen-1.11.2/lib/
sqreen/backported_queue.rb:24: [BUG] pthread_mutex_lock: Invalid argument (EINVAL)
2018-03-21T22:53:41.973454+00:00 app[web.2]: ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linu
x]
2018-03-21T22:53:41.973456+00:00 app[web.2]:
2018-03-21T22:53:41.973457+00:00 app[web.2]: -- Control frame information -----------------------
----------------------
2018-03-21T22:53:41.973462+00:00 app[web.2]: c:0057 p:---- s:0407 e:000406 CFUNC  :signal
2018-03-21T22:53:41.973466+00:00 app[web.2]: c:0056 p:0017 s:0403 e:000402 BLOCK  /app/vendor/bund
le/ruby/2.5.0/gems/sqreen-1.11.2/lib/sqreen/backported_queue.rb:24 [FINISH]
2018-03-21T22:53:41.973473+00:00 app[web.2]: c:0055 p:---- s:0400 e:000399 CFUNC  :synchronize
2018-03-21T22:53:41.973481+00:00 app[web.2]: c:0054 p:0008 s:0396 e:000395 BLOCK  /app/vendor/bund
```

```
le/ruby/2.5.0/gems/sqreen-1.11.2/lib/sqreen/backported_queue.rb:22 [FINISH]
2018-03-21T22:53:41.973489+00:00 app[web.2]: c:0053 p:---- s:0393 e:000392 CFUNC  :handle_interrup
t
2018-03-21T22:53:41.973502+00:00 app[web.2]: c:0052 p:0022 s:0388 e:000387 METHOD /app/vendor/bund
le/ruby/2.5.0/gems/sqreen-1.11.2/lib/sqreen/backported_queue.rb:21
2018-03-21T22:53:41.973507+00:00 app[web.2]: c:0051 p:0031 s:0383 e:000382 METHOD /app/vendor/bund
le/ruby/2.5.0/gems/sqreen-1.11.2/lib/sqreen/capped_queue.rb:22
```

While putting my ideas in order when writing this bug report I finally managed to find a way to reproduce the issue in a simple Ruby script. Please find it attached with matching Ruby trace report.

Nota this script does *not* crash in the following Ruby version on my computer:
ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-linux]
ruby 2.4.3p205 (2017-12-14 revision 61247) [x86_64-linux]

but does crash under:
ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linux]

Let me know if I can help further

Finally, thanks for writing & maintaining such an awesome language !

Related issues:

Related to Ruby - Bug #15383: Reproducible crash: crash.rb:6: [BUG] unexpecte...	**Closed**

Associated revisions

Revision a2d63ea2fb84c962abddae877e9493fc57cfce1a - 03/27/2018 09:28 AM - Eric Wong

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]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@62934 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision a2d63ea2 - 03/27/2018 09:28 AM - Eric Wong

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]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@62934 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b456eab2ea77eda51c8d5c06c24d195a6a2932e1 - 04/20/2018 03:22 AM - Eric Wong

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]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63210 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b456eab2 - 04/20/2018 03:22 AM - Eric Wong

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]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63210 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision af72dcd91b70e94b0f8299ab0464dafe884d2695 - 04/20/2018 10:53 PM - Eric Wong

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

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63215 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision af72dcd9 - 04/20/2018 10:53 PM - Eric Wong

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

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63215 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision d2f52a5fb7835c299d0b769e2d0423d80c522fc8 - 04/30/2018 11:47 PM - 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]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63309 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

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

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63309 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63d9ab33fe419167382fc03be2c00413a78c05cb - 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_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
      [ruby-core:86316] [Bug #14634]

    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]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@66912 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63d9ab33fe419167382fc03be2c00413a78c05cb - 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_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
      [ruby-core:86316] [Bug #14634]

    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]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@66912 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

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

    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]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@66912 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 703d9f61f0057be889ae0e468e0031781f450e3b - 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]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@67259 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 703d9f61f0057be889ae0e468e0031781f450e3b - 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]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@67259 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 703d9f61 - 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]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@67259 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 03/27/2018 01:24 AM - zetaben (Benoit Larroque)

- Subject changed from Queue#push seems to crash to Queue#push seems to crash after fork

#2 - 03/27/2018 04:21 AM - normalperson (Eric Wong)

benoit@sqreen.io wrote:

> https://bugs.ruby-lang.org/issues/14634

This seems identical to [Bug #14578] (sorry, my fault)
Can you try r62852 in ruby_2_5 branch (or r62668 in trunk)?
Thanks.

#3 - 03/27/2018 06:49 AM - zetaben (Benoit Larroque)

- File rev62852.txt added

Hello Eric,

I just tried compiling & running with r62852 on my computer. I still have the same issue:

```
/versatile/queue2.rb:62: [BUG] pthread_mutex_lock: Invalid argument (EINVAL)
ruby 2.5.0p43 (2018-03-19 revision 62852) [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0003 p:---- s:0014 e:000013 CFUNC  :push
```

(please find full trace attached)

Looking a bit through the issues earlier, I actually found that https://bugs.ruby-lang.org/issues/14474 was having very similar symptoms as my issue but the offending test seemed not really related (threads only no queue). I forgot to mention this in my initial report, sorry!

#4 - 03/27/2018 08:51 AM - normalperson (Eric Wong)

benoit@sqreen.io wrote:

> I just tried compiling & running with r62852 on my computer. I still have the same issue:
>
> ```
> /versatile/queue2.rb:62: [BUG] pthread_mutex_lock: Invalid argument (EINVAL)
> ruby 2.5.0p43 (2018-03-19 revision 62852) [x86_64-linux]
> ```

Thanks. Still my bug :< Fix coming in a few.

> Looking a bit through the issues earlier, I actually found
> that https://bugs.ruby-lang.org/issues/14474 was having very
> similar symptoms as my issue but the offending test seemed not
> really related (threads only no queue). I forgot to mention
> this in my initial report, sorry!

Ah, thanks. I wonder if autoload cause that.... Will check in a
bit.

#5 - 03/27/2018 09:27 AM - normalperson (Eric Wong)

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

#6 - 03/27/2018 09:28 AM - Anonymous

- Status changed from Open to Closed

Applied in changeset trunk|r62934.

---

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]

#7 - 03/27/2018 09:31 AM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> benoit@sqreen.io wrote:
>
> > I just tried compiling & running with r62852 on my computer. I still have the same issue:
> >
> > ```
> > /versatile/queue2.rb:62: [BUG] pthread_mutex_lock: Invalid argument (EINVAL)
> > ruby 2.5.0p43 (2018-03-19 revision 62852) [x86_64-linux]
> > ```
>
> Thanks. Still my bug :< Fix coming in a few.

Can you try r62934? Thanks.

> > Looking a bit through the issues earlier, I actually found
> > that https://bugs.ruby-lang.org/issues/14474 was having very
> > similar symptoms as my issue but the offending test seemed not
> > really related (threads only no queue). I forgot to mention
> > this in my initial report, sorry!
>
> Ah, thanks. I wonder if autoload cause that.... Will check in a
> bit.

Ugh, tired and I still have other stuff to do :< I guess it's
less urgent

#8 - 03/27/2018 09:55 PM - zetaben (Benoit Larroque)

```
$ ./tool/runruby.rb  --version
ruby 2.6.0dev (2018-03-27 trunk 62934) [x86_64-linux]
$ ./tool/runruby.rb ~/AgentRuby/bin/queue2.rb
8640.47011093644200: Starting
8640.47011093738780: Consumer parent thread 47011093739080
8640.47011093644200: Forking
8645.47011093708060: Consumer thread 47011093739080
8645.47011093644200: finished child
8640.47011093644200: finished parent
```

That seem to have fixed it! Newbie question, do you plan to backport it to 2.5.x series also ?

Thanks for the awesome response time!

#9 - 03/28/2018 08:21 AM - normalperson (Eric Wong)

benoit@sqreen.io wrote:

> That seem to have fixed it! Newbie question, do you plan to backport it to 2.5.x series also ?

Thanks for the confirmation. I'm not too happy about the memory
increase for such objects; but I will work on that next month.

naruse handles the backporting for 2.5 branch.

> Thanks for the awesome response time!

No prob, just lucky since I've been offline a lot, lately.

#10 - 03/28/2018 09:52 PM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> benoit@sqreen.io wrote:
>
> > That seem to have fixed it! Newbie question, do you plan to backport it to 2.5.x series also ?
>
> Thanks for the confirmation. I'm not too happy about the memory
> increase for such objects; but I will work on that next month.
>
> naruse handles the backporting for 2.5 branch.

I guess we missed the merge window for 2.5.1

Fwiw, I consider doing fork after creating threads to be
dangerous and bug-prone in any language. AFAIK, pthreads
implementations do not support it, even.

Ruby currently can support this because of GVL; but I'm not sure
how well we can support this pattern in the future.

Just to reiterate:

```
Thread.new; fork;  # unsafe
fork; Thread.new;  # safe
```

For the fork+exec case, we use vfork+exec via Process.spawn to
ensure backtick and system() work inside Threads. (Similarly,
POSIX has posix_spawn(3)).

#11 - 03/29/2018 04:37 AM - zetaben (Benoit Larroque)

> I guess we missed the merge window for 2.5.1
> Too bad! next time one then !
>
> Fwiw, I consider doing fork after creating threads to be
> dangerous and bug-prone in any language. AFAIK, pthreads
> implementations do not support it, even.
>
> Ruby currently can support this because of GVL; but I'm not sure
> how well we can support this pattern in the future.
>
> Just to reiterate:
>
> Thread.new; fork; # unsafe
> fork; Thread.new; # safe
>
> For the fork+exec case, we use vfork+exec via Process.spawn to
> ensure backtick and system() work inside Threads. (Similarly,
> POSIX has posix_spawn(3)).

Thanks for the detailed infos ! I'll see if I can tune our design with this in mind.

#12 - 03/31/2018 02:16 AM - zetaben (Benoit Larroque)

- File cons_defined_sigsev.txt added

Hello @normalperson (Eric Wong),

The customer just sent me another crasher report (segfault). It happens in another location in Ruby on Module.const_defined?. I looked at the ruby code while going through the crasher trace, and it seems that it's using a very similar pattern (wait list + rb_ensure) near line 2192 in rb_autoload_load (on trunk).
From what I got from the r62934 commit, the rb_ensure pattern might here also not be sufficient after a fork.

I'm sorry though, I have no reproduction script yet.
This happens on a newly updated Ruby 2.5.1

I attached the crasher log. To my non-expert eye, it really felt like the same bug happening in another place, but if you want me to open another dedicated ticket I'll be happy to.

#13 - 04/02/2018 07:33 AM - normalperson (Eric Wong)

Thanks. I suspected autoload was a problem; too :x I'll try to
take a look at it in the coming days (sorry, many things going
on...) If you can start with a small test case that would be
much appreciated. Thanks again.

But really, forking after creating threads is a terrible idea
(outside of Process.spawn/system/backtick)

#14 - 04/20/2018 03:32 AM - normalperson (Eric Wong)

benoit@sqreen.io wrote:

> The customer just sent me another crasher report (segfault).
> It happens in another location in Ruby on
> Module.const_defined?. I looked at the ruby code while going
> through the crasher trace, and it seems that it's using a very
> similar pattern (wait list + rb_ensure) near line 2192 in
> rb_autoload_load (on trunk).

Sorry for the delay, can you try r63210? Curious if somebody
else hit this and we failed to notice, since it's been a bug
since 2.3.0 in 2015.

(also at: https://80x24.org/spew/20180420032210.663-1-e@80x24.org/raw)

#15 - 06/26/2018 01:26 PM - jmgarnier (Jean-Michel Garnier)

- Status changed from Closed to Open

- ruby -v changed from ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linux] to ruby 2.5.1p57 (2018-03-29 revision 63029) [x86_64-linux]

I also encountered segfaults when upgrading from ruby 2.4 to 2.5.1
they happen in https://github.com/codegram/futuroscope gem which initalizes a pool of threads at startup. When passenger smart spawning forks the process, it segfaults.

I tested the same code with ruby 2.6 preview and no segfaults.

Any idea when naruse will backport to 2.5 branch please?

We are not in hurry, we can also wait for 2.6.

#16 - 07/25/2018 05:54 PM - tenderlovemaking (Aaron Patterson)

Hi, we've been hitting this bug at work too. I tested these commits (backported them on to Ruby 2.5) and they seem to clear up the issue. Naruse, could we have these backported to the ruby_2_5 branch? I made backport commits here: https://github.com/github/ruby/pull/40

If there's anything I can do to help, please let me know.

Thanks!

#17 - 07/28/2018 04:23 PM - shan (Shannon Skipper)

Hi, we've also been running into this at work. Thanks for the fix and backport! We look forward to a 2.5.2 release.

#18 - 12/06/2018 12:10 AM - nagachika (Tomoyuki Chikanaga)

- Related to Bug #15383: Reproducible crash: crash.rb:6: [BUG] unexpected THREAD_KILLED added

#19 - 12/06/2018 12:12 AM - nagachika (Tomoyuki Chikanaga)

- Status changed from Open to Closed

#20 - 01/23/2019 02:14 PM - nagachika (Tomoyuki Chikanaga)

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

ruby_2_5 r66912 merged revision(s) 62934,63210,63215,63309.

Files

einval.stripped.txt	59.9 KB	03/26/2018	zetaben (Benoit Larroque)
sigsegv.stripped.txt	60.8 KB	03/26/2018	zetaben (Benoit Larroque)
queue2.rb	1.15 KB	03/26/2018	zetaben (Benoit Larroque)
repro.trace.txt	12.6 KB	03/26/2018	zetaben (Benoit Larroque)
rev62852.txt	10.3 KB	03/27/2018	zetaben (Benoit Larroque)
cons_defined_sigsev.txt	5.42 KB	03/31/2018	zetaben (Benoit Larroque)