Ruby master - Bug #5258
SizedQueue  Bug #5195

09/01/2011 10:23 PM - Glass_saga (Masaki Matsushita)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>kosaki (Motohiro KOSAKI)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 1.9.4dev (2011-09-01 trunk 33157) [x86_64-linux]</td>
</tr>
</tbody>
</table>

Backport:

**Description**

```ruby
require 'thread'
sq = SizedQueue.new(1)
sq.push(0)
t1 = Thread.start { sq.push(1) ; sleep }
nil until t1.stop?
t1.wakeup
nil until t1.stop?
t2 = Thread.start { sq.push(2) }
nil until t1.stop? && t2.stop?
p t1, t2
sq.instance_eval{ p @queue_wait }
3.times{ sq.pop }
t2.join
```

上記のコードを実行すると、

```ruby
#/ #
#/ /usr/local/lib/ruby/1.9.1/thread.rb:185:in sleep': deadlock detected (fatal)
from /usr/local/lib/ruby/1.9.1/thread.rb:185:in block in pop'
from internal/prelude:10:in synchronize'
from /usr/local/lib/ruby/1.9.1/thread.rb:180:in pop'
from /usr/local/lib/ruby/1.9.1/thread.rb:324:in pop'
from sized_queue.rb:19:inblock in '
from sized_queue.rb:19:in times'
from sized_queue.rb:19:in
```  

関連issue:

Related to Ruby master - Bug #5195: Queue#popでsleepしているthreadをwakeupさせるとQueue...
Closed 08/16/2011

Related to Ruby master - Bug #5355: Sync_mにBug #5195やBug #5258と同様のバグ
Closed 09/23/2011
Associated revisions

Revision 7198053a - 09/09/2012 12:32 PM - kosaki (Motohiro KOSAKI)

- lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

- lib/thread.rb (SizedQueue#push): ditto.

- test/thread/test_queue.rb (test_sized_queue_and_wakeup,
  test_queue_pop_interrupt, test_sized_queue_pop_interrupt,
  test_sized_queue_push_interrupt): new tests.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@36938 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 36938 - 09/09/2012 12:32 PM - kosaki (Motohiro KOSAKI)

- lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

- lib/thread.rb (SizedQueue#push): ditto.

- test/thread/test_queue.rb (test_sized_queue_and_wakeup,
  test_queue_pop_interrupt, test_sized_queue_pop_interrupt,
  test_sized_queue_push_interrupt): new tests.

Revision 36938 - 09/09/2012 12:32 PM - kosaki (Motohiro KOSAKI)

- lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

- lib/thread.rb (SizedQueue#push): ditto.

- test/thread/test_queue.rb (test_sized_queue_and_wakeup,
  test_queue_pop_interrupt, test_sized_queue_pop_interrupt,
  test_sized_queue_push_interrupt): new tests.

Revision 36938 - 09/09/2012 12:32 PM - kosaki (Motohiro KOSAKI)

- lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

- lib/thread.rb (SizedQueue#push): ditto.

- test/thread/test_queue.rb (test_sized_queue_and_wakeup,
  test_queue_pop_interrupt, test_sized_queue_pop_interrupt,
  test_sized_queue_push_interrupt): new tests.

Revision 36938 - 09/09/2012 12:32 PM - kosaki (Motohiro KOSAKI)

- lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

- lib/thread.rb (SizedQueue#push): ditto.

- test/thread/test_queue.rb (test_sized_queue_and_wakeup,
  test_queue_pop_interrupt, test_sized_queue_pop_interrupt,
  test_sized_queue_push_interrupt): new tests.

Revision 36938 - 09/09/2012 12:32 PM - kosaki (Motohiro KOSAKI)

- lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

- lib/thread.rb (SizedQueue#push): ditto.

- test/thread/test_queue.rb (test_sized_queue_and_wakeup,
  test_queue_pop_interrupt, test_sized_queue_pop_interrupt,
  test_sized_queue_push_interrupt): new tests.
• lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

• lib/thread.rb (SizedQueue#push): ditto.


Revision 36938 - 09/09/2012 12:32 PM - kosaki (Motohiro KOSAKI)

• lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]

• lib/thread.rb (SizedQueue#push): ditto.


History

#1 - 09/02/2011 09:26 PM - kosaki (Motohiro KOSAKI)
- Status changed from Open to Assigned
- Assignee set to kosaki (Motohiro KOSAKI)

#2 - 09/03/2011 08:08 PM - nagachika (Tomoyuki Chikanaga)
- File queue.patch added

#3 - 04/11/2012 07:43 PM - rklemme (Robert Klemme)
This is also present in 1.9.3 and there is an even simpler test case:

$ ruby19 -r thread -e 'q=SizedQueue.new 10;1_000_000.times {|i| p i;q.enq i}'
0
1
2
3
4
5
6
7
8
9
10
/out/lib/ruby/1.9.1/thread.rb:301:in sleep': deadlock detected (fatal)
from /out/lib/ruby/1.9.1/thread.rb:301:inblock in push'
from internal:prelude:10:in synchronize'
from /out/lib/ruby/1.9.1/thread.rb:297:inpush'
from -e:1:intimes'
from -e:1:in `'

rklemme@padrklemme2 /cygdrive/c/SCMws/RKlemme/JavaProducts_oslee_ngcp_dev_R3.3_be4rb
$ ruby19 -v
ruby 1.9.3p125 (2012-02-16) [i386-cygwin]

#4 - 04/11/2012 08:09 PM - nobu (Nobuyoshi Nakada)
It seems natural because the only thread is about to sleep.

#5 - 04/11/2012 11:37 PM - rklemme (Robert Klemme)
nobu (Nobuyoshi Nakada) wrote:

It seems natural because the only thread is about to sleep.

I would expect the thread to block indefinitely. Signalling an error here seems to try to be too smart. Even in absence of other threads I can imagine conditions under which data is read from the queue (for example a signal handler).

But even if an error is signalled here, it is certainly not a deadlock - for that you need at least two threads. This is rather something like "only blocking thread is suspended (with no chance to wake up)". Maybe that error should be controllable via a switch, e.g.

Thread.single_thread_block_is_error = true # default false

#6 - 04/12/2012 12:29 AM - mame (Yusuke Endoh)
Hello,

2012/4/11 rklemme (Robert Klemme) shortcutter@googlemail.com:

nobu (Nobuyoshi Nakada) wrote:

It seems natural because the only thread is about to sleep.

I would expect the thread to block indefinitely. Signalling an error here seems to try to be too smart. Even in absence of other threads I can imagine conditions under which data is read from the queue (for example a signal handler).

Interesting. The error is indeed "false positive." I did not noticed signal handler.

But I'm not enthusiastic for your proposal. It is too conservative. It will increase too many "false negative."

But even if an error is signalled here, it is certainly not a deadlock - for that you need at least two threads. This is rather something like "only blocking thread is suspended (with no chance to wake up)". Maybe that error should be controllable via a switch, e.g.

Thread.single_thread_block_is_error = true # default false

Do you have any practical case where you get bothered by the "false positive"? If so, please open a feature request ticket with the use case.

--
Yusuke Endoh mame@tsg.ne.jp

#7 - 04/12/2012 04:49 PM - rklemme (Robert Klemme)
Hi again,

mame (Yusuke Endoh) wrote:

2012/4/11 rklemme (Robert Klemme) shortcutter@googlemail.com:

nobu (Nobuyoshi Nakada) wrote:

It seems natural because the only thread is about to sleep.

I would expect the thread to block indefinitely. Signalling an error here seems to try to be too smart. Even in absence of other threads I can imagine conditions under which data is read from the queue (for example a signal handler).

Interesting. The error is indeed "false positive." I did not noticed signal handler.

I was lucky because I was just sensitized by Eric's recent article about signal handling in Ruby. :-)  

But I'm not enthusiastic for your proposal. It is too conservative. It will increase too many "false negative."

You mean processes with no unblocked threads would go unnoticed? What about the effort for JRuby and others to mimic this behavior?
But even if an error is signaled here, it is certainly not a deadlock - for that you need at least two threads. This is rather something like "only blocking thread is suspended (with no chance to wake up)". Maybe that error should be controllable via a switch, e.g.

Thread.single_thread_block_is_error = true # default false

Do you have any practical case where you get bothered by the "false positive"?
If so, please open a feature request ticket with the use case.

Not at the moment. The case I thought I had evaporated because it was a "silent thread death" case (i.e. Thread.abort_on_exception was at default - false - and the only other thread had silently died where I thought it would still be running). But a better error message than "deadlock" as I have suggested above would have made me find the source of the issue much more quickly because it had hinted at the real problem.

I think I'll open a feature request to at least change the error message in case there is no other thread left. Btw, is this really a deadlock detection or does the code only check for blocked threads? From the code in check_deadlock_i() it is not totally obvious to me, I think the check only ensures at least one live thread - at least it does not seem to try to find the deadlock loop (as e.g. some relational databases do).

Kind regards
robert

#8 - 09/09/2012 09:32 PM - kosaki (Motohiro KOSAKI)
- Status changed from Assigned to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r36938.
Masaki, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- lib/thread.rb (Queue#pop): Fixed double registration issue when mutex.sleep is interrupted. [Bug #5258] [ruby-dev:44448]
- lib/thread.rb (SizedQueue#push): ditto.
- test/thread/test_queue.rb (test_sized_queue_and_wakeup,
  test_queue_pop_interrupt, test_sized_queue_pop_interrupt,
  test_sized_queue_push_interrupt): new tests.

Files
patch.diff 399 Bytes 09/01/2011 Glass_saga (Masaki Matsushita)
queue.patch 2.35 KB 09/03/2011 nagachika (Tomoyuki Chikanaga)