Ruby master - Bug #4266

Timeouts in threads cause "ThreadError: deadlock; recursive locking"

01/12/2011 01:59 AM - cjbottaro (Christopher Bottaro)

Status:	Closed		
Priority:	Normal		
Assignee:	kosaki (Motohiro KOSAKI)		
Target version:	2.0.0		
ruby -v:	ruby 1.9.2p136 (2010-12-25 revision 30365) [x86_64-darwin10.6.0]	**Backport:**	

Description

=begin
Run the attached file (or this pastie http://pastie.org/1448542) a few times and you'll eventually get:

ThreadError: deadlock; recursive locking:
internal:prelude:8:in lock'
<internal:prelude>:8:insynchronize'
bin/deadlock_test.rb:86:in block (4 levels) in <main>'
/Users/cjbottaro/.rvm/rubies/ruby-1.9.2-p136/lib/ruby/1.9.1/timeout.rb:57:intimeout'
bin/deadlock_test.rb:85:in block (3 levels) in <main>'
bin/deadlock_test.rb:83:intimes'
bin/deadlock_test.rb:83:in `block (2 levels) in '

I've had the script run successfully over 5 times in a row before getting the errors, so if it doesn't happen the first few times... keep trying.

The problem doesn't happen in 1.8.7 or Jruby, but does happen in 1.9.1.
=end

Related issues:

Related to Ruby master - Bug #4285: Ruby don't have asynchrounous exception s...	**Closed**	**01/17/2011**
Related to Ruby master - Bug #4289: Timeouts in threads cause SEGV	**Closed**	**01/18/2011**

Associated revisions

Revision 6c56dae4 - 11/19/2012 10:22 AM - kosaki (Motohiro KOSAKI)

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37724 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37724 - 11/19/2012 10:22 AM - kosaki (Motohiro KOSAKI)

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

Revision 37724 - 11/19/2012 10:22 AM - kosaki (Motohiro KOSAKI)

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

Revision 37724 - 11/19/2012 10:22 AM - kosaki (Motohiro KOSAKI)

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

Revision 37724 - 11/19/2012 10:22 AM - kosaki (Motohiro KOSAKI)

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

Revision 37724 - 11/19/2012 10:22 AM - kosaki (Motohiro KOSAKI)

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

Revision 37724 - 11/19/2012 10:22 AM - kosaki (Motohiro KOSAKI)

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

History

#1 - 01/12/2011 02:49 AM - headius (Charles Nutter)

=begin
FYI, JRuby does not use timeout.rb anymore (as of JRuby 1.3ish I think). We use a native implementation based on Java's timed, thread-pooling executor in java.util.concurrent. The original timeout has many race conditions that are very difficult to fix.
=end

#2 - 01/12/2011 03:02 AM - headius (Charles Nutter)

=begin
A question was raised about whether #to should understand context, as in knowing a top-level object being coerced to YAML should include the --- header. The answer seems to come from Marshal.

Marshal.dump(object) knows how to produce the marshal header, linking, and so on. Marshal is the master of this format. But Marshal defers to the objects themselves for actual *content* to go into that marshaled output, calling _dump or marshal_dump on each object in turn. With the #to protocol, Marshal.dump could be implemented as:

def Marshal.dump(obj)
emit_header
obj.to(Marshal)
end

In the same way that marshal_dump is used today.

Another concern raised is whether #to should be expected on all objects all the time, rather than just always calling MyClass.coerce(obj). The answer is simple: you want individual source types to control how they coerce to a target type, rather than expecting the target type to know about all possible sources. The default protocol where #to calls type.coerce would simply be a default behavior.
=end

#3 - 01/12/2011 03:11 AM - headius (Charles Nutter)

=begin
Sorry about that, wrong bug. Too many bug windows open.
=end

#4 - 01/12/2011 03:11 AM - headius (Charles Nutter)

=begin
Sorry about that, wrong bug. Too many bug windows open.
=end

#5 - 01/12/2011 03:41 AM - kosaki (Motohiro KOSAKI)

=begin
This test program also makes ruby crash on my envronment (trunk + linux x86_64).

Related [Bug #3880]?

internal:prelude:8: [BUG] Segmentation fault
ruby 1.9.3dev (2011-01-09 trunk 30500) [x86_64-linux]

-- Control frame information ---
c:0011 p:---- s:0036 b:0036 l:000035 d:000035 CFUNC :exception
c:0010 p:---- s:0034 b:0034 l:000033 d:000033 CFUNC :lock
c:0009 p:0011 s:0031 b:0031 l:000030 d:000030 METHOD internal:prelude:8
c:0008 p:0011 s:0028 b:0028 l:000bb8 d:000027 BLOCK deadlock_test.rb:21
c:0007 p:0109 s:0026 b:0026 l:001ad0 d:001ad0 METHOD /usr/local/lib/ruby/1.9.1/timeout.rb:57
c:0006 p:0019 s:0014 b:0014 l:000bb8 d:002580 BLOCK deadlock_test.rb:20
c:0005 p:---- s:0011 b:0011 l:000010 d:000010 FINISH
c:0004 p:---- s:0009 b:0009 l:000008 d:000008 CFUNC :times
c:0003 p:0010 s:0006 b:0006 l:000bb8 d:0012c8 BLOCK deadlock_test.rb:18
c:0002 p:---- s:0004 b:0004 l:000003 d:000003 FINISH

```
c:0001 p:---- s:0002 b:0002 l:000001 d:000001 TOP

-- Ruby level backtrace information ----------------------------------------
deadlock_test.rb:18:in block (2 levels) in <main>'
deadlock_test.rb:18:intimes'
deadlock_test.rb:20:in block (3 levels) in <main>'
/usr/local/lib/ruby/1.9.1/timeout.rb:57:intimeout'
deadlock_test.rb:21:in block (4 levels) in <main>'
<internal:prelude>:8:insynchronize'
internal:prelude:8:in lock'
<internal:prelude>:8:inexception'

-- C level backtrace information -------------------------------------------
./ruby() [0x51ff25] vm_dump.c:797
./ruby() [0x564296] error.c:249
./ruby(rb_bug+0xb1) [0x564431] error.c:266
./ruby() [0x4aec70] signal.c:624
/lib64/libpthread.so.0() [0x382140f440]
./ruby(st_lookup+0xe) [0x4b63ce] st.c:326
./ruby() [0x50d3af] vm_method.c:402
./ruby(rb_funcall2+0x2e) [0x51ccee] vm_eval.c:227
./ruby(rb_class_new_instance+0x30) [0x44f490] object.c:1570
./ruby() [0x516e33] vm_eval.c:79
./ruby() [0x517304] vm_eval.c:290
./ruby() [0x417691] eval.c:564
./ruby(rb_exc_raise+0x26) [0x417886] eval.c:473
./ruby() [0x52487a] thread.c:1301
./ruby(rb_mutex_lock+0x44a) [0x527cfa] thread.c:3266
./ruby() [0x50f0b7] vm_insnhelper.c:403
./ruby() [0x510a36] insns.def:1010
./ruby() [0x5157eb] vm.c:1150
./ruby(rb_yield+0x66) [0x51e116] vm.c:591
./ruby() [0x4469d1] numeric.c:3217
./ruby() [0x50f0b7] vm_insnhelper.c:403
./ruby() [0x510a36] insns.def:1010
./ruby() [0x5157eb] vm.c:1150
./ruby() [0x5186df] vm.c:607
./ruby() [0x5272d1] thread.c:453
./ruby() [0x5273e0] thread_pthread.c:522
/lib64/libpthread.so.0() [0x3821407761]
/lib64/libc.so.6(clone+0x6d) [0x3820ce151d]

=end
```

#6 - 01/12/2011 02:41 PM - kosaki (Motohiro KOSAKI)

=begin
Hm, the crash issue seems different and unrelated one from this. becuase ruby_1_9_2 can reproduce recursive deadlock issue, but can't reproduce crash.

=end

#7 - 01/12/2011 02:45 PM - kosaki (Motohiro KOSAKI)

- File diffdiff added

- Priority changed from Normal to 5

=begin
Old days, C# had similar issue. and they changed Monitor.enter(= our Mutex.lock) semantics.

http://www.bluebytesoftware.com/blog/2007/01/30/MonitorEnterThreadAbortsAndOrphanedLocks.aspx

So, Can we take similar way? The attached patch kill a race in mutex.synchronize. At least, my ruby_1_9_2 branch don't reproduce the issue anymore.

Thanks.
=end

#8 - 01/12/2011 03:01 PM - kosaki (Motohiro KOSAKI)

- Assignee set to ko1 (Koichi Sasada)

=begin
And, I think asynchronous exception should be blocked and delayed while running ensure block.

Otherwise Mutex.unlock can be bypassed.

ko1, What do you think?
=end

#9 - 01/12/2011 03:18 PM - kosaki (Motohiro KOSAKI)

=begin
That said, our Mutex.synchronize() method has two unsafe point.

```
def synchronize
self.lock
// (1)
begin
yield
ensure
// (2)
self.unlock rescue nil
end
end
```

If the mutex got thread#raise at (1), it doesn't call unlock because
it haven't entered -begin- block. And if got at (2), it also doesn't call
unlock because we are no longer in -begin- block.

My patch fixed only (1).

Thanks.
=end

#10 - 01/12/2011 04:45 PM - kosaki (Motohiro KOSAKI)

- File mutex-synchronize-use-c-implementation.patch added

=begin
Alternative one is here. This one fixes both (1) and (2). But it only fixes Mutex#synchronize. That said, timeout{} still might not call ruby level ensure block.

=end

#11 - 01/18/2011 04:58 PM - kosaki (Motohiro KOSAKI)

=begin
Hi,

I plan to commit mutex-synchronize-use-c-implementation.patch at this weekend. So if anyone have objection, please let me know soon.

=end

#12 - 03/26/2011 10:25 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#13 - 04/06/2011 07:42 AM - normalperson (Eric Wong)

=begin
I have a short-term fix for the related issue in Bug #4289
http://redmine.ruby-lang.org/attachments/1572/0001-timeout.rb-avoid-introducing-new-class-for-every-tim.patch

It works both with and without the mutex-synchronize-use-c-implementation.patch
posted here.

=end

#14 - 04/27/2011 04:02 AM - briangug (Brian Gugliemetti)

=begin
This issue also applies to MonitorMixin (monitor.rb) as it uses the same lock/begin yield/ensure block.
=end

#15 - 05/05/2011 09:01 AM - briangug (Brian Gugliemetti)

- File monitor-synchronize-use-c-implementation.patch added

#16 - 06/26/2011 04:11 PM - nahi (Hiroshi Nakamura)

- Target version changed from 1.9.2 to 1.9.3

#17 - 06/26/2011 07:10 PM - ko1 (Koichi Sasada)

- Target version changed from 1.9.3 to 2.0.0

Let's talk it after 1.9.3 release.

#18 - 08/13/2011 08:55 PM - jtara (Jon Tara)

Brian Gugliemetti wrote:

> File monitor-synchronize-use-c-implementation.patch added

There is a parameter mis-match between the declaration and definition of rb_mon_synchronize. Compilation fails on at least OSX.

#19 - 04/19/2012 08:13 AM - MingVonsalis (Ming Vonsalis)

jtara (Jon Tara) wrote:

> Brian Gugliemetti wrote:
>
> > File monitor-synchronize-use-c-implementation.patch added
>
> There is a parameter mis-match between the declaration and definition of rb_mon_synchronize. Compilation fails on at least OSX.

http://originsofinuitart.posterous.com/

#20 - 11/17/2012 12:43 AM - trapni (Christian Parpart)

We just ran into this very ugly bug in production, too. We would really appreciate a fix in 1.9.4, a pure bugfix release of the 1.9 branch since this is a major bug. :)

#21 - 11/17/2012 01:04 AM - ko1 (Koichi Sasada)

- Assignee changed from ko1 (Koichi Sasada) to kosaki (Motohiro KOSAKI)

I passed this ticket to thread specialist.

#22 - 11/17/2012 10:31 AM - kosaki (Motohiro KOSAKI)

Brian,

Can you please provide us a reproducer for you monitor.rb issue? And can you please make a new ticket for your issue to prevent cross line discussion? I plan to close this ticket soon and I also hope to don't forget your issue.

Thanks.

#23 - 11/19/2012 07:22 PM - kosaki (Motohiro KOSAKI)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r37724.
Christopher, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- prelude.rb: Moved Mutex#synchronize to
- thread.c (rb_mutex_synchronize_m): here. [Bug #4266]

Files

deadlock_test.rb	554 Bytes	01/12/2011	cjbottaro (Christopher Bottaro)
diffdiff	1.38 KB	01/12/2011	kosaki (Motohiro KOSAKI)
mutex-synchronize-use-c-implementation.patch	1.28 KB	01/12/2011	kosaki (Motohiro KOSAKI)

monitor-synchronize-use-c-implementation.patch	4.6 KB	05/05/2011	briangug (Brian Gugliemetti)