Ruby master - Bug #6634
Deadlock with join and ConditionVariable
06/23/2012 11:49 PM - meh. (meh. I don't care)

Status: Rejected
Priority: Normal
Assignee: mame (Yusuke Endoh)
Target version: 2.0.0
ruby -v: ruby 1.9.3p194 (2012-04-20 revision 35410) [x86_64-linux]

Description
I'm getting a fatal deadlock in one of my gems, it's a simple threadpool implementation.
The library works both in Rubinius and JRuby, so I guess it's a bug.
The gem is here: [https://github.com/meh/ruby-threadpool](https://github.com/meh/ruby-threadpool)
The example that crashes is attached.
Basically it raises a fatal deadlock if you join a thread and then call ConditionVariable#wait, I'm not 100% sure if the bug is in the ConditionVariable or what, all I know is that it happens in that situation and that it works on Rubinius and JRuby.

History
#1 - 06/24/2012 05:54 AM - Anonymous
- File noname added

On Sat, Jun 23, 2012 at 11:49:14PM +0900, meh. (meh. I don't care) wrote:

Issue #6634 has been reported by meh. (meh. I don't care).

Bug #6634: Deadlock with join and ConditionVariable
[https://bugs.ruby-lang.org/issues/6634](https://bugs.ruby-lang.org/issues/6634)

Author: meh. (meh. I don't care)
Status: Open
Priority: Normal
Assignee: mame (Yusuke Endoh)
Category: core
Target version: 2.0.0
ruby -v: ruby 1.9.3p194 (2012-04-20 revision 35410) [x86_64-linux]

I'm getting a fatal deadlock in one of my gems, it's a simple threadpool implementation.
The library works both in Rubinius and JRuby, so I guess it's a bug.
The gem is here: [https://github.com/meh/ruby-threadpool](https://github.com/meh/ruby-threadpool)
The example that crashes is attached.
Basically it raises a fatal deadlock if you join a thread and then call ConditionVariable#wait, I'm not 100% sure if the bug is in the ConditionVariable or what, all I know is that it happens in that situation and that it works on Rubinius and JRuby.

I can't seem to reproduce this error:
[http://www.youtube.com/watch?v=8J_eBXZ7ud4](http://www.youtube.com/watch?v=8J_eBXZ7ud4)

Can you reduce the error to a self contained example that reliably fails?

--
Aaron Patterson
Always happens, on Arch Linux x86_64.

```ruby
ruby reduced.rb
reduced.rb:13:in join': deadlock detected (fatal)
from reduced.rb:13:in'
```

This is true deadlock. The above thread.join has no chance to exit successfully. Can you please elaborate your point?

Then I can't come up with a reduced testcase, I know that it triggers a fatal deadlock in my gem when it's actually not a deadlock.

It works both in JRuby and Rubinius.

Unfortunately, we don't have an esp capability. "The library works both in Rubinius and JRuby, so I guess it's a bug." don't gave me any hint. sorry.

The issue is that it's thinking it's deadlocking when actually another thread is going to shutdown the threadpool (hence broadcasting on the cond and not being a deadlock).

I succeeded to reproduce the issue, by adding set_trace_func to lol.rb, redirecting the output to the file, and repeating the invocation until the error occurs. It looks very very timing sensitive issue.

```
$ gem install threadpool

$ ./ruby -v
ruby 2.0.0dev (2012-11-05 trunk 37474) [x86_64-linux]

$ ruby -e 'loop { system("./ruby lol2.rb > t") || break }'
I reviewed the source of threadpool gem, but I could find no problem. Precisely, it may attempt to call undefined method named "reason"; it is clearly irrelevant.

Kosaki-san, could you try to reproduce? The core behavior looks to me indeed strange (too subtle to explain in English, sorry), but I failed to find the bug.

面倒なので日本語で。

再現性が乏しく（うちの環境で100回実行に1回くらい？）gdbを使いこなせないのでprintf debugで戦ってみたんですが、確かにcoreが怪しい挙動をしている気がしました。

CV内のmutexをlockしたはずなのになぜかthreadpool内のmutexがlockされているような、そうでないような。

大物のタイミングバグの予感がする（GC issueかも知れませんが）のですが、小崎さんの環境で再現できたら勝利だと思うので、試してみてもらえますでしょうか。

--
Yusuke Endoh mame@tsg.ne.jp

#10 - 11/07/2012 11:06 AM - kosaki (Motohiro KOSAKI)
- Assignee changed from kosaki (Motohiro KOSAKI) to ko1 (Koichi Sasada)

#11 - 11/07/2012 12:22 PM - kosaki (Motohiro KOSAKI)
Hi mame-san,

ko1 found the second case (i.e. below) is a his regression since October. He told me he plan to fix soon.

$ ruby -e 'loop { system("./ruby lol2.rb > t") || break }'
<internal:prelude>:8:in 'lock': deadlock; recursive locking [ThreadError]
  from <internal:prelude>:8:in 'synchronize'
  from /home/mame/work/local/lib/ruby/2.0.0/thread.rb:69:in 'wait'
  from /home/mame/work/local/lib/ruby/gems/2.0.0/gems/threadpool-0.1.2/lib/threadpool.rb:234:in `block (3 levels) in spawn_thread'
  from <internal:prelude>:10:in 'synchronize'
  from /home/mame/work/local/lib/ruby/gems/2.0.0/gems/threadpool-0.1.2/lib/threadpool.rb:222:in `block (2 levels) in spawn_thread'
  from /home/mame/work/local/lib/ruby/gems/2.0.0/gems/threadpool-0.1.2/lib/threadpool.rb:249:in `loop'
  from /home/mame/work/local/lib/ruby/gems/2.0.0/gems/threadpool-0.1.2/lib/threadpool.rb:249:in `block in spawn_thread'

And I couldn't reproduce this issue at commit r37074 (Oct 3). So I think we haven't reproduce an original issue yet.

#12 - 11/13/2012 06:59 PM - ko1 (Koichi Sasada)
- Assignee changed from ko1 (Koichi Sasada) to mame (Yusuke Endoh)

Maybe this second problem is fixed at r37647. mame-san, could you check it?

#13 - 11/13/2012 08:11 PM - mame (Yusuke Endoh)
- Status changed from Assigned to Feedback

Worked. Thank you!

Then, anyone can reproduce the original problem? Meh, can you still reproduce?

--
Yusuke Endoh mame@tsg.ne.jp

#14 - 02/08/2013 11:57 PM - mame (Yusuke Endoh)
- Status changed from Feedback to Rejected

Marking this as rejected due to lack of feedback by the submitter.
Hi there,

I've faced similar problem with ruby 2.0.0p0 (2013-02-24 revision 39474) x86_64-darwin12.1.0

Please checkout my attached code. Let me know if I could help you more. Or if i'm doing something dumb :)

nhm tanveeer hossain khan wrote:

Hi there,

I've faced similar problem with ruby 2.0.0p0 (2013-02-24 revision 39474) x86_64-darwin12.1.0

Hey, I have the same problem. I took the test case you posted, reduced it further, and fiddled with the numbers of threads, etc. See attached. It crashed reliably for me, always right after launching it.

If we are using Ruby threads the wrong way, please let us know. If not, could you please take another look at this issue and possibly reactivate it?

Thanks.

For the record, the test case is malformed. Bummer. I think the one I based it on (from khan) is malformed as well. My apologies if you spent time on it.

NP :)}