Backport187 - Backport #1471

"Mutual join" deadlock detection faulty in 1.8.6 and 1.8.7

05/15/2009 05:19 PM - JohnCarter (John Carter)

Status:	Closed	
Priority:	Normal	
Assignee:	shyouhei (Shyouhei Urabe)	

Description

=begin
If a third thread is involved, the code in eval.c in function
rb_thread_join0 to detect a mutual join dead lock is faulty.

This effects ruby 1.8.6 p368 and ruby 1.8.7 p 160 but not ruby 1.9.1 p
129

What seems to be happening is a sequence like this....

main thread grabs mutex

thread two blocks waiting for mutex

thread one blocks waiting for mutex and th->join is set to the main thread.

the main thread unlocks, which wakes thread two. Thread two now has mutex, thread one is still blocked.

the main thread attempts to join thread one, rb_thread_join0 notes
that thread one is in a WAIT_JOIN waitfor state AND th->join still
points at the main thread and (erroneously) reports deadlock.

ruby-1.8.7-p160/eval.c==========
rb_thread_join0(th, limit)
rb_thread_t th;
double limit;
{
enum rb_thread_status last_status = THREAD_RUNNABLE;

```
 if (rb_thread_critical) rb_thread_deadlock();
 if (!rb_thread_dead(th)) {
     if (th == curr_thread)
         rb_raise(rb_eThreadError, "thread 0x%lx tried to join itself",
                  th->thread);
     if ((th->wait_for & WAIT_JOIN) && th->join == curr_thread)
         rb_raise(rb_eThreadError, "Thread#join: deadlock 0x%lx - mutual join(0x%lx)",
                  curr_thread->thread, th->thread);
```

==========
Affected versions.

/opt/ruby/ruby-1.8.6-p368/bin/ruby -v
ruby 1.8.6 (2009-03-31 patchlevel 368) [i686-linux]

/opt/ruby/ruby-1.8.7-p160/bin/ruby -v
ruby 1.8.7 (2009-04-08 patchlevel 160) [i686-linux]

Here is a test script to trigger it....
=mutual_join_bug.rb==========
require 'thread'

m = Mutex.new
m.lock
wt2 = Thread.new do
m.lock
sleep 2

m.unlock
end

Ensure wt2 is waiting on m
sleep 0.1

wt1 = Thread.new do
m.lock
m.unlock
end
Ensure wt1 is waiting on m
sleep 0.1

Give it to wt2
m.unlock

wt1.join
==========

Output...
/opt/ruby/ruby-1.8.7-p160/bin/ruby -w mutual_join_bug.rb mutual_join_bug.rb:24:in `join': Thread#join: deadlock 0xb7daa1bc - mutual join(0xb7d9bc48) (ThreadError)
from mutual_join_bug.rb:24

The following patch "fixes" it...
==========
--- eval.c_orig 2009-03-23 22:28:31.000000000 +1300
+++ eval.c 2009-05-15 15:15:34.141270568 +1200
@@ -11426,9 +11426,13 @@
if (th == curr_thread)
rb_raise(rb_eThreadError, "thread 0x%lx tried to join itself",
th->thread);
+/* This test for mutual join deadlock is faulty in that

- it doesn't allow for the possibility that a third thread
- may now hold the mutex. if ((th->wait_for & WAIT_JOIN) && th->join == curr_thread) rb_raise(rb_eThreadError, "Thread#join: deadlock 0x%lx - mutual join(0x%lx)", curr_thread->thread, th->thread); +*/ if (curr_thread->status == THREAD_TO_KILL) last_status = THREAD_TO_KILL; if (limit == 0) return Qfalse; ==========

But I'm not sure it is the correct fix. I'm still experience some
strangeness which I haven't pinned down yet.

I would seem to me that the correct fix is when a mutex is unlocked,
and then locked by some other thread, the th->join pointers of all
threads waiting on that mutex should be updated to point at the thread
now holding the mutex.
=end

History

#1 - 05/15/2009 05:37 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
Applied in changeset r23432.
=end

#2 - 05/15/2009 05:39 PM - nobu (Nobuyoshi Nakada)

- Category set to core

- Status changed from Closed to Assigned

- Assignee set to shyouhei (Shyouhei Urabe)

=begin

=end

#3 - 05/15/2009 05:58 PM - rue (Eero Saynatkari)

=begin
Sidenote--

Excerpts from message of Fri May 15 11:19:51 +0300 2009:

> Here is a test script to trigger it....
> =mutual_join_bug.rb==========
> require 'thread'
>
> m = Mutex.new
> m.lock
> wt2 = Thread.new do
> m.lock
> sleep 2
> m.unlock
> end
>
> # Ensure wt2 is waiting on m
>
> sleep 0.1

The analysis looks correct, but usually you would not want
to rely solely on a sleep here, but also check the #status
before proceeding.

--
Magic is insufficiently advanced technology.

=end

#4 - 05/16/2009 04:56 PM - brent (Brent Roman)

=begin

I don't dispute the analysis either, but...
I cannot seem to duplicate this bug in 1.6.8, 1.8.6-p287 or 1.8.7-p72

Is it specific to just the patchlevels mentioned?
Is it likely to be platform specific?
(I'm testing on x86 32-bin Linux)

When I run the mutual_join_bug script, it consistently returns without
outputting anything.

- brent

Eric Hodel-2 wrote:

> Bug #1471: "Mutual join" deadlock detection faulty in 1.8.6 and 1.8.7
> http://redmine.ruby-lang.org/issues/show/1471
>
> Author: John Carter
> Status: Open, Priority: Normal
> Category: core
> ruby -v: ruby 1.8.7 (2009-04-08 patchlevel 160) [i686-linux]
>
> If a third thread is involved, the code in eval.c in function
> rb_thread_join0 to detect a mutual join dead lock is faulty.
>
> This effects ruby 1.8.6 p368 and ruby 1.8.7 p 160 but not ruby 1.9.1 p
> 129

--
View this message in context:
http://www.nabble.com/-ruby-core%3A23457---Bug--1471--%22Mutual-join%22-deadlock-detection-faulty-in-1.8.6-and-1.8.7-tp23555444p23571265.html
Sent from the ruby-core mailing list archive at Nabble.com.

=end

#5 - 05/16/2009 05:20 PM - nobu (Nobuyoshi Nakada)

=begin
Hi,

At Sat, 16 May 2009 16:56:23 +0900,
Brent Roman wrote in [ruby-core:23479]:

> Is it specific to just the patchlevels mentioned?

1.8.6-p364 and 1.8.7-p152.

--
Nobu Nakada

=end

#6 - 05/18/2009 02:13 PM - JohnCarter (John Carter)

=begin
Thanks for the fix...

Unfortunately I think it introduces another bug.

Your patch add invokes rb_thread_set_join from adjust_join from wake_one from unlock_mutex_inner.

Alas, wake_one is also invoked from signal_condvar which is invoked from signal_condvar_call from rb_condvar_signal. Thus all the join pointer of all threads waiting on that conditionvariable will be updated to point at the first thread on the condvar waiting list. Which assumes, perhaps over boldly, that the woken thread will be the one to get the lock.

Now the thread that has just been woken should have be at a ConditionVariable.wait(mutex) point, having come through rb_condvar_wait->wait_condvar->ensure(wait_list,,lock_mutex_call) which will do the right thing.

ie. The call to adjust_join should be moved out of "wake_one" and into "unlock_mutex_inner"

Edited to remove the following which I got wrong.....
IGNORE THIS LINE->The other place that's going to invoke adjust_join, but shouldn't, is wake_all.
=end

#7 - 05/18/2009 02:40 PM - JohnCarter (John Carter)

=begin
Here is an alternate patch for ext/thread.c relative to ruby-1.8.7-p160...
--- ext/thread/thread.c.orig 2009-05-18 17:31:50.225602484 +1200
+++ ext/thread/thread.c 2009-05-18 17:32:38.689286319 +1200
@@ -205,6 +205,16 @@
return ary;
}

+static void
+adjust_join(const List *list, VALUE new)
+{

- extern void rb_thread_set_join _((VALUE, VALUE));
- Entry *entry;
- for (entry = list->entries; entry; entry = entry->next) {
- rb_thread_set_join(entry->value, new);

- }
  +}
  +
  static VALUE
  wake_thread(VALUE thread)
  {
  @@ -471,6 +481,11 @@
  }

  waking = wake_one(&mutex->waiting);
  +

- if (!NIL_P(waking)) {

- adjust_join(&mutex->waiting, waking);
- }
  +
  mutex->owner = waking;

  return waking;

Even with the fix I'm still getting a deadlock. I will have to look at
it somemore tomorrow.

=end

#8 - 05/19/2009 02:39 PM - JohnCarter (John Carter)

- File ruby_1_8_7_reset_waitfor_on_dead_threads.patch added

- File ruby_1_8_7_thread_adjust_join.patch added

=begin
Ok, That fix wasn't sufficient. There were still other deadlock issues, which I believe I have now fixed.

In eval.c in rb_thread_schedule we have the following code handling the case of a thread we are waiting to join that has died....

```
if (th->wait_for & WAIT_JOIN) {
    if (rb_thread_dead(th->join)) {
    th->status = THREAD_RUNNABLE;
    found = 1;
    }
```

Note that we have updated the th->status, but we have left the th->wait_for unchanged! Adding...
th->wait_for = 0;
resolved the problem.

In the new function rb_thread_set_join that Nobu added...
http://svn.ruby-lang.org/cgi-bin/viewvc.cgi/branches/ruby_1_8/eval.c?r1=23316&r2=23432
he sets th->wait_for to WAIT_JOIN.

I see this as a precondition for this function to be called.

Attached is two patches relative to 1.8.7-p160

=end

#9 - 05/21/2009 08:24 AM - nobu (Nobuyoshi Nakada)

=begin
Hi,

At Tue, 19 May 2009 14:39:21 +0900,
John Carter wrote in [ruby-core:23500]:

> Attached is two patches relative to 1.8.7-p160

Couldn't you make a patch against the 1.8 head?

--
Nobu Nakada

=end

#10 - 06/21/2009 03:52 PM - hansdegraaff (Hans de Graaff)

=begin
The patches from comment #8 apply cleanly against 1.8.7-p174 and they do solve the threading issues I have with capistrano. Thanks John!

We will be adding a revision bump of ruby 1.8.7 with these patches in Gentoo shortly.
=end

#11 - 07/09/2009 02:16 PM - hansdegraaff (Hans de Graaff)

=begin
Unfortunatey it seems I was too optimistic in my previous comment. Even with the patches I'm still seeing problems with threading when deploying

with capistrano. Consequently we haven't added these to Gentoo just yet.

Bug http://redmine.ruby-lang.org/issues/show/1484 is a duplicate of this bug.
=end

#12 - 09/03/2009 02:42 PM - dazuma (Daniel Azuma)

=begin
@Hans: I do not believe http://redmine.ruby-lang.org/issues/show/1484 is a duplicate of this bug. 1484 generally manifests through capistrano, and it was traced, by the capistrano people (see https://capistrano.lighthouseapp.com/projects/8716/tickets/79-capistrano-hangs-on-shell-command-for-many-computers-on-ruby-186-p368) and independently by me, to an issue with calling IO#select in multiple threads. That issue (1484) has, as far as I can tell, been resolved on the 1.8 head (see http://redmine.ruby-lang.org/issues/show/1993) and is currently assigned for backport to 1.8.7. This current issue (1471) seems to be a distinct issue. So John Carter's patches may be sufficient for this bug.
=end

#13 - 07/17/2010 04:47 PM - tad (Tadashi Saito)

=begin
Hi all,

My friend told me that this bug seems to be still alive.
I confirmed it in above version:

ruby 1.8.7 (2010-06-23 patchlevel 299) [i686-linux]

I checked that ruby_1_8 was modified, but it's never backported to
ruby_1_8_7. Shyouhei, could you backport this please?

In addition, IMHO, to change bug status "Closed" was not a good
thing because this bug is opened as a bug of 1.8.7, not 1.8.

--
Tadashi Saito
=end

#14 - 07/17/2010 04:50 PM - tad (Tadashi Saito)

=begin
Oops, I was wrong.
This is correctly "Assigned." Sorry for misunderstanding.
=end

#15 - 02/21/2011 06:41 PM - hongli (Hongli Lai)

=begin
This bug is still very much alive in Ruby 1.8.7-p334. mutual_join_bug.rb still triggers the bug. Though it is fixed in 1.9.

John Carter's patches fixes the problem in mutual_join_bug.rb, but unfortunately causes a regression. Here's a test case:
https://gist.github.com/836861
This test case spawns 50 threads. Each thread repeatedly downloads from microsoft.com, then grabs a lock, increases a counter and prints a progress message. If you let this script run for a few seconds, then interrupt it with Ctrl-C then sometimes it will crash with "[BUG] Internal consistency failure!". Sample session:

$ ruby threaddownload.rb
80287834: Progress = 1
808b4a68: Progress = 2
808b35a0: Progress = 3
808b29fc: Progress = 4
808b1e30: Progress = 5
808735f4: Progress = 6
Cthreaddownload.rb:19: [BUG] Internal consistency failure! Expected thread to already be in waiting to join state 8, was in 0

This makes threading on Ruby 1.8 currently nearly useless. I don't have enough expertise to solve this problem unfortunately but I would like to request the core team to put more priority on this bug.
=end

#16 - 02/22/2011 08:46 PM - hongli (Hongli Lai)

=begin
It appears that applying this patch instead of John Carter's patches fixes both the mututal join problem as well as the crashing problems:
https://github.com/ruby/ruby/commit/072ccff04debff32072e771bc078fe8bc14ccaad

But it's only applied to the 1_8 branch. If the fix is correct I think it should be backported to 1.8.7 ASAP.
=end

#17 - 02/22/2011 10:39 PM - shyouhei (Shyouhei Urabe)

=begin
Hmm, I see. I will backport it.
=end

#18 - 12/21/2016 05:14 AM - shyouhei (Shyouhei Urabe)

- Status changed from Assigned to Closed

Files

mutual_join_bug.rb	261 Bytes	05/15/2009	JohnCarter (John Carter)
ruby_1_8_7_reset_waitfor_on_dead_threads.patch	1.13 KB	05/19/2009	JohnCarter (John Carter)
ruby_1_8_7_thread_adjust_join.patch	737 Bytes	05/19/2009	JohnCarter (John Carter)