[PATCH] thread.c (blocking_region_end): clear ubf before unregister_ubf_list

07/28/2018 12:45 AM - normalperson (Eric Wong)

<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></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td></td>
</tr>
<tr>
<td>Backport:</td>
<td>2.3: REQUIRED, 2.4: REQUIRED, 2.5: REQUIRED</td>
</tr>
</tbody>
</table>

Description
thread.c (blocking_region_end): clear ubf before unregister_ubf_list

If we keep ubf set after unregistering, there is a window for other threads (including timer thread) to put this thread back on the ubf_list right away. Entering ubf_list unexpectedly after GVL acquisition may cause spurious wakeup and trigger unexpected behavior.

Finally, clear ubf before acquiring GVL, to since ubf is useless during GVL acquisition anyways and we don't want to waste cycles in other threads calling ubf for useless work.

I found this bug while rewriting GVL to handle timer-thread duty and eliminate separate timer thread.

Associated revisions
Revision 856bd77a - 07/28/2018 03:10 AM - normal
thread.c (blocking_region_end): clear ubf before unregister_ubf_list

If we keep ubf set after unregistering, there is a window for other threads (including timer thread) to put this thread back on the ubf_list right away. Entering ubf_list unexpectedly after GVL acquisition may cause spurious wakeup and trigger unexpected behavior.

Finally, clear ubf before acquiring GVL, to since ubf is useless during GVL acquisition anyways and we don't want to waste cycles in other threads calling ubf for useless work.

[ruby-core:88141] [Bug #14945]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64083 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 64083 - 07/28/2018 03:10 AM - normalperson (Eric Wong)
thread.c (blocking_region_end): clear ubf before unregister_ubf_list

If we keep ubf set after unregistering, there is a window for other threads (including timer thread) to put this thread back on the ubf_list right away. Entering ubf_list unexpectedly after GVL acquisition may cause spurious wakeup and trigger unexpected behavior.

Finally, clear ubf before acquiring GVL, to since ubf is useless during GVL acquisition anyways and we don't want to waste cycles in other threads calling ubf for useless work.

[ruby-core:88141] [Bug #14945]

Revision 64083 - 07/28/2018 03:10 AM - normal
thread.c (blocking_region_end): clear ubf before unregister_ubf_list

If we keep ubf set after unregistering, there is a window for
other threads (including timer thread) to put this thread back on the ubf_list right away. Entering ubf_list unexpectedly after GVL acquisition may cause spurious wakeup and trigger unexpected behavior.

Finally, clear ubf before acquiring GVL, to since ubf is useless during GVL acquisition anyways and we don’t want to waste cycles in other threads calling ubf for useless work.

[ruby-core:88141] [Bug #14945]

Revision 86d35a6b - 07/30/2018 09:48 PM - normal
thread_pthread.c (unregister_ubf_list): assert unblock.func is unset

We must not allow reentry into ubf_list_head once we delete ourselves, otherwise we could hang in there forever.

[ruby-core:88218] [Bug #14945]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64134 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 64134 - 07/30/2018 09:48 PM - normalperson (Eric Wong)

thread_pthread.c (unregister_ubf_list): assert unblock.func is unset

We must not allow reentry into ubf_list_head once we delete ourselves, otherwise we could hang in there forever.

[ruby-core:88218] [Bug #14945]

History

#1 - 07/28/2018 03:10 AM - normalperson (Eric Wong)
- Status changed from Open to Closed

Applied in changeset trunk|r64083.

thread_c (blocking_region_end): clear ubf before unregister_ubf_list

If we keep ubf set after unregistering, there is a window for other threads (including timer thread) to put this thread back on the ubf_list right away. Entering ubf_list unexpectedly after GVL acquisition may cause spurious wakeup and trigger unexpected behavior.

Finally, clear ubf before acquiring GVL, to since ubf is useless during GVL acquisition anyways and we don’t want to waste cycles in other threads calling ubf for useless work.

[ruby-core:88141] [Bug #14945]

#2 - 07/30/2018 09:42 PM - normalperson (Eric Wong)

https://bugs.ruby-lang.org/issues/14945

Actually, r64083 may not be completely safe, either, as the list_empty checks are dangerous, at least w/o GVL.

However, this order must be enforced:
Working on fixes...

#3 - 08/08/2018 05:51 AM - ko1 (Koichi Sasada)
:+1:

One off topic question. list_empty() is thread-safe?

#4 - 08/08/2018 05:53 AM - ko1 (Koichi Sasada)

ko1 (Koichi Sasada) wrote:

One off topic question. list_empty() is thread-safe?

Sorry it should be safe (I misread as other operation. sorry).

#5 - 08/08/2018 09:33 AM - normalperson (Eric Wong)

ko1@atdot.net wrote:

ko1 (Koichi Sasada) wrote:

One off topic question. list_empty() is thread-safe?

Sorry it should be safe (I misread as other operation. sorry).

Right, good question, though... for register_ubf_list
and unregister_ubf_list, they are safe and I can commit
below patch to clarify.

However, list_empty in ubf_threads_empty I'm not 100% sure
about...

diff --git a/thread_pthread.c b/thread_pthread.c
index 29805ef2df..fbeb0c1d1 100644
--- a/thread_pthread.c
+++ b/thread_pthread.c
@@ -1125,6 +1125,15 @@ register_ubf_list(rb_thread_t *th)
      struct list_node *node = &th->native_thread_data.ubf_list;
      /*
       * list_empty check is safe here without ubf_list_lock held
+      * because th->interrupt_lock is already held by
+      * rb_threadptr_interrupt_common.
+      *
+      * rb_threadptr_interrupt_common
+      * ubf_select (== th->unblock.func)
+      * register_ubf_list (this function)
+      */
      if (!list_empty((struct list_head*)node)) {
        rb_native_mutex_lock(&ubf_list_lock);
        list_add(&ubf_list_head, node);
@@ -1141,6 +1150,11 @@ unregister_ubf_list(rb_thread_t *th)
      /* we can't allow re-entry into ubf_list_head */
      VM_ASSERT(th->unblock.func == 0);
      }
+      /*
+      * list_empty check is safe here without ubf_list_lock held
+      * because we already cleared th->unblock.func while
+      * th->interrupt_lock was held.
+      */
      if (!list_empty((struct list_head*)node)) {
        rb_native_mutex_lock(&ubf_list_lock);
        list_del_init(node);
      }