Ruby - Bug #1181

[BUG] thread_free: keeping_mutexes must be NULL

02/20/2009 05:10 AM - orem (Chris Schlaeger)

Status:	Closed		
Priority:	Normal		
Assignee:	mame (Yusuke Endoh)		
Target version:			
ruby -v:	ruby 1.9.2dev (2009-02-15 trunk 22332) [x86_64-linux]	**Backport:**	

Description

=begin
I've tried to run taskjuggler with ruby 1.9 CVS from today and got internal error messages. The applications runs fine with ruby 1.8.7. I tried to cut it down to a smaller test case, but it seems like you need the full app to trigger the problem. When run in uniprocessor mode, it works fine even with 1.9. But when running in SMP mode (-c 2 or larger number) it multithreads and forks to run on muliple CPUs. The problem seems to be thread related as the error message indicates. The locations in the ruby code seem to be random and change with every run. Here are the steps to reproduce the problem:

1. Download the taskjuggler gem: http://www.taskjuggler.org/tj3/taskjuggler-0.0.1.gem and install it
2. change to the 'examples' that comes with the gem or download acso.tjp from http://www.taskjuggler.org/tj3/examples/acso.tjp
3. tj3 -c 2 acso.tjp
 =end

History

#1 - 02/20/2009 12:39 PM - mame (Yusuke Endoh)

=begin
HI,

2009/2/20 Chris Schlaeger redmine@ruby-lang.org:

> I've tried to run taskjuggler with ruby 1.9 CVS from today and got internal error messages. The applications runs fine with ruby 1.8.7. I tried to cut it down to a smaller test case, but it seems like you need the full app to trigger the problem. When run in uniprocessor mode, it works fine even with 1.9. But when running in SMP mode (-c 2 or larger number) it multithreads and forks to run on muliple CPUs. The problem seems to be thread related as the error message indicates. The locations in the ruby code seem to be random and change with every run. Here are the steps to reproduce the problem:
>
> 1. Download the taskjuggler gem: http://www.taskjuggler.org/tj3/taskjuggler-0.0.1.gem and install it
> 2. change to the 'examples' that comes with the gem or download acso.tjp from http://www.taskjuggler.org/tj3/examples/acso.tjp
> 3. tj3 -c 2 acso.tjp

Thank you for your detailed report!
I confirmed the issue. I'll try to tackle it tonight.

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#2 - 02/20/2009 12:45 PM - shyouhei (Shyouhei Urabe)

- Assignee set to mame (Yusuke Endoh)

=begin

=end

#3 - 02/20/2009 09:10 PM - mame (Yusuke Endoh)

=begin
Hi,

2009/2/20 Yusuke ENDOH mame@tsg.ne.jp:

> HI,

> 2009/2/20 Chris Schlaeger redmine@ruby-lang.org:
>
>> I've tried to run taskjuggler with ruby 1.9 CVS from today and got internal error messages. The applications runs fine with ruby 1.8.7. I tried to cut it down to a smaller test case, but it seems like you need the full app to trigger the problem. When run in uniprocessor mode, it works fine even with 1.9. But when running in SMP mode (-c 2 or larger number) it multithreads and forks to run on muliple CPUs. The problem seems to be thread related as the error message indicates. The locations in the ruby code seem to be random and change with every run. Here are the steps to reproduce the problem:
>>
>> 1. Download the taskjuggler gem: http://www.taskjuggler.org/tj3/taskjuggler-0.0.1.gem and install it
>> 2. change to the 'examples' that comes with the gem or download acso.tjp from http://www.taskjuggler.org/tj3/examples/acso.tjp
>> 3. tj3 -c 2 acso.tjp
>
> Thank you for your detailed report!
> I confirmed the issue. I'll try to tackle it tonight.

The following patch should fix the issue. Could you try?

Index: bootstraptest/test_thread.rb

--- bootstraptest/test_thread.rb (revision 22466)
+++ bootstraptest/test_thread.rb (working copy)
@@ -216,6 +216,18 @@
end
}

+assert_equal 'ok', %{

- open("zzz.rb", "w") do |f|
- f.puts <<-END

- [redacted]

- [redacted]

- [redacted]

- END
- end
- require "zzz.rb"
- $result
  +}

assert_finish 3, %{
th = Thread.new {sleep 2}
th.join(1)
Index: thread.c

--- thread.c (revision 22466)
+++ thread.c (working copy)
@@ -291,7 +291,7 @@
struct rb_mutex_struct *next_mutex;
} mutex_t;

-static void rb_mutex_unlock_all(mutex_t *mutex);
+static void rb_mutex_unlock_all(mutex_t *mutex, rb_thread_t *th);

void
rb_thread_terminate_all(void)
@@ -305,7 +305,7 @@

```
 /* unlock all locking mutexes */
 if (th->keeping_mutexes) {
```

- rb_mutex_unlock_all(th->keeping_mutexes);

- rb_mutex_unlock_all(th->keeping_mutexes, GET_THREAD());
  }

  thread_debug("rb_thread_terminate_all (main thread: %p)\n", (void *)th);
  @@ -339,6 +339,12 @@
  thread_cleanup_func(void *th_ptr)
  {
  rb_thread_t *th = th_ptr;

- /* unlock all locking mutexes */

- if (th->keeping_mutexes) {

- rb_mutex_unlock_all(th->keeping_mutexes, th);

- th->keeping_mutexes = NULL;

- }
  thread_cleanup_func_before_exec(th_ptr);
  native_thread_destroy(th);
  }
  @@ -434,12 +440,6 @@
  (void *)th, th->locking_mutex);
  }

- /* unlock all locking mutexes */

- if (th->keeping_mutexes) {

- [redacted]

- }

- /* delete self other than main thread from living_threads */
  if (th != main_th) {
  st_delete_wrap(th->vm->living_threads, th->self);
  @@ -457,7 +457,6 @@
  }
  join_th = join_th->join_list_next;
  }

- if (th != main_th) rb_check_deadlock(th->vm);

  if (!th->root_fiber) {
  rb_thread_recycle_stack_release(th->stack);
  @@ -465,6 +464,7 @@
  }
  }
  thread_cleanup_func(th);

- if (th != main_th) rb_check_deadlock(th->vm);
  if (th->vm->main_thread == th) {
  ruby_cleanup(state);
  }
  @@ -3146,7 +3146,7 @@
  }

static void
-rb_mutex_unlock_all(mutex_t *mutexes)
+rb_mutex_unlock_all(mutex_t *mutexes, rb_thread_t *th)
{
const char *err;
mutex_t *mutex;*
@@ -3156,7 +3156,7 @@
/ rb_warn("mutex #<%p> remains to be locked by terminated thread",
mutexes); */
mutexes = mutex->next_mutex;

- err = mutex_unlock(mutex, GET_THREAD());

- err = mutex_unlock(mutex, th);
  if (err) rb_bug("invalid keeping_mutexes: %s", err);
  }
  }

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#4 - 02/23/2009 04:35 PM - orem (Chris Schlaeger)

=begin
I can confirm that your patch fixes the problem. Thanks for your quick help!

Chris

=end

#5 - 02/23/2009 04:57 PM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:22292] Re: [Bug #1181] [BUG] thread_free: keeping_mutexes must be NULL"
on Fri, 20 Feb 2009 21:09:38 +0900, Yusuke ENDOH mame@tsg.ne.jp writes:

|The following patch should fix the issue. Could you try?

Please check in.

```
        matz.
```

=end

#6 - 02/24/2009 01:27 AM - mame (Yusuke Endoh)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
Applied in changeset r22577.
=end

Files

log	72.6 KB	02/20/2009	orem (Chris Schlaeger)