Ruby - Bug #2629

ConditionVariable#wait(mutex, timeout) should return whether the condition was signalled, not the waited time

01/22/2010 11:04 PM - hongli (Hongli Lai)

Status:	Rejected		
Priority:	Normal		
Assignee:	mame (Yusuke Endoh)		
Target version:	1.9.2		
ruby -v:	ruby 1.9.2dev (2010-01-21 trunk 26367) [x86_64-darwin10.2.0]	**Backport:**	

Description

=begin
At this time, ConditionVariable#wait on 1.9.2dev works as follows:

ConditionVariable#wait(mutex, timeout) => integer
Waits for at most 'timeout' time. 'timeout' may be a floating
point number.
Returns the number of *seconds* spent waiting. So even if it
actually spent 3.5 seconds waiting, it'll either return 3 or 4.
There is no way to check whether the wait was signaled or
timed out, you have to guess based on the time waited.

Ideally it should just return true or false, just like how JRuby's version behaves:

ConditionVariable#wait(mutex, timeout) => boolean
Waits for at most 'timeout' time. 'timeout' may be a floating
point number.
Returns true if condition was signaled, false if it timed out.

Rationale

I have an application which uses ConditionVariable#wait(mutex, timeout) in order to wait on a condition for a bounded time. The use case is as follows:

```
There is a background thread which performs some cleaning function every x seconds. We also want t
o be able to tell the thread to clean now, or to exit (i.e. quitting its main loop so that we can
join the thread). This thread waits on a condition variable for x seconds, and then checks whether
 there was a timeout on the wait, or whether the wait as signaled. In case of the former it will r
un the cleanup code. In case of the latter it'll check whether @quit is set, and then either stop
the loop or run the cleanup code.
```

The problem with 1.9.2dev's return value is that there is no way for me to detect whether the condition was signaled or whether the wait timed out.
I can try to guess whether there was a timeout by checking whether the waited time is >= the timeout, but the return value doesn't allow me to do even that because it has seconds resolution. Right now I have to work around it by placing Time.now.to_f timers around the #wait call, so that I can measure the difference in time in miliseconds or microseconds.

Please make ConditionVariable#wait(mutex, timeout) behave like JRuby's version.
=end

History

#1 - 03/17/2010 02:28 AM - mame (Yusuke Endoh)

- Category set to YARV

- Target version set to 1.9.2

=begin
Hi, Hongli Lai

2010/1/22 Hongli Lai redmine@ruby-lang.org:

> ? There is a background thread which performs some cleaning function every x seconds. We also want to be able to tell the thread to clean now, or to exit (i.e. quitting its main loop so that we can join the thread). This thread waits on a condition variable for x seconds, and then checks whether there was a timeout on the wait, or whether the wait as signaled. In case of the former it will run the cleanup code. In case of the latter it'll check whether @quit is set, and then either stop the loop or run the cleanup code.

Could you show me a concrete code?
I guess it is more robust to always check @quit, regardless of whether
it was timed out or signaled.

I could be wrong because I'm not familiar with multi-thread programming,
but I doubt that it is potentially dangerous to depend on the reason of
wait termination. Couldn't it cause race condition?

I have no idea about concrete race scenario, but there is a circumstance
where Java's Object#wait seems not to return the termination reason.

> The problem with 1.9.2dev's return value is that there is no way for me to detect whether the condition was signaled or whether the wait timed out.
> I can try to guess whether there was a timeout by checking whether the waited time is >= the timeout, but the return value doesn't allow me to do even that because it has seconds resolution. Right now I have to work around it by placing Time.now.to_f timers around the #wait call, so that I can measure the difference in time in miliseconds or microseconds.

FYI, ConditionVariable#wait returns self currently, for 1.8 compatibility.

--
Yusuke ENDOH mame@tsg.ne.jp
=end

#2 - 03/20/2010 03:20 AM - mame (Yusuke Endoh)

=begin
Hi,

2010/3/17 Yusuke Endoh redmine@ruby-lang.org:

> I could be wrong because I'm not familiar with multi-thread programming,
> but I doubt that it is potentially dangerous to depend on the reason of
> wait termination. Couldn't it cause race condition?
>
> I have no idea about concrete race scenario, but there is a circumstance
> where Java's Object#wait seems not to return the termination reason.

I found ptherad_cond_timedwait returns ETIMEOUT.
My concern seemed to be needless fear.

But I still doubt a little whether your code has a race:

http://www.opengroup.org/onlinepubs/000095399/functions/pthread_cond_timedwait.html

> Similarly, when pthread_cond_timedwait() returns with the timeout error,
> the associated predicate may be true due to an unavoidable race between
> the expiration of the timeout and the predicate state change.
>
> The application needs to recheck the predicate on any return because it
> cannot be sure there is another thread waiting on the thread to handle
> the signal, and if there is not then the signal is lost. The burden is
> on the application to check the predicate.

And I realized a bigger problem from the document; it is considered as
a race to speficy a relative time as timeout.

ConditionVariable#wait should accept an absolute time, at least.
But it needs Mutex#sleep accept Time instead of Integer. Umm.

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#3 - 03/20/2010 08:11 PM - hongli (Hongli Lai)

=begin
I guess you are right.
=end

#4 - 05/04/2010 04:01 AM - mame (Yusuke Endoh)

- Category changed from YARV to lib

- Assignee set to mame (Yusuke Endoh)

=begin
Hi,

I suggest temporal revert of timeout argument of ConditionVariable#wait.
As I said in [ruby-core:28803], the API design has a potential race.

Unfortunately, 1.9.2 is already frozen for release. We should discuss
about the API towards 1.9.3, including return value and argument format.

Unless there is no objection, I'll revert r25058.

--
Yusuke Endoh mame@tsg.ne.jp
=end

#5 - 05/11/2010 03:55 AM - mame (Yusuke Endoh)

- Status changed from Open to Rejected

=begin
Hi, Hongli

Even if there is no return value, you can detect the timeout
without guessing by the following:

```
mutex.synchronize do
t = Time.now + x
until @quit
t2 = Time.now
cv.wait(mutex, t - t2) if t > t2
break if @quit
if Time.now > t
t += x
# cleanup code
end
end
# stop the loop
end
```

I recommend this code because it is robust against spurious
wakeup.

I think the feature you requested can be used to make the
code more efficient, like the following:

```
mutex.synchronize do
t = Time.now + x
until @quit
t2 = Time.now
timeout = t <= t2 || cv.wait(mutex, t - t2)
break if @quit  # (A)
if timeout
t += x
# cleanup code
end
end
# stop the loop
end
```

But, the feature is not mandatory, and is even prone to misuse
(the check (A) is very easy to forget).

In addition, Tanaka Akira, who has added the timeout feature
to CV#wait, rejected the feature.

Consequently, I close this ticket. Sorry for decision against
your exception, and also sorry for not deciding sooner.

--
Yusuke Endoh mame@tsg.ne.jp
=end