Ruby - Bug #2505

Threads behave inconsistently across platforms.

12/20/2009 02:08 PM - docwhat (Christian Höltje)

Status:	Rejected		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:			
ruby -v:	1.9.1-p376	**Backport:**	

Description

=begin
In Ruby 1.9, Thread is now using native threads. The problem with this is that threading models don't work the same across different platforms.

Some examples:
priority doesn't do the same thing on Solaris than Linux.
loop{} will lock the process up in Solaris, but not Linux.

There are plenty of other examples.

In addition, the fact that native threads use the same class name is also confusing. It means that code that would work on fully GIL encumbered green threads would work fine but may break in mysterious ways using native threads.

Finally, there is the fact that any code that could possibly be called from a native thread now has to be thread safe, making writing C extensions difficult.
=end

History

#1 - 12/20/2009 07:00 PM - naruse (Yui NARUSE)

=begin

> Some examples:
> priority doesn't do the same thing on Solaris than Linux.
> loop{} will lock the process up in Solaris, but not Linux.

They may true. If a difference cause a practical problem, please report it.

> In addition, the fact that native threads use the same class name is also confusing.
> It means that code that would work on fully GIL encumbered green threads would work fine
> but may break in mysterious ways using native threads.
>
> Finally, there is the fact that any code that could possibly be called from
> a native thread now has to be thread safe, making writing C extensions difficult.

These two are wrong.
Ruby 1.9 uses native threads but still use GIL just because of the problem you pointing out.
=end

#2 - 12/21/2009 01:58 AM - docwhat (Christian Höltje)

=begin

> > Some examples:
> > priority doesn't do the same thing on Solaris than Linux.
> > loop{} will lock the process up in Solaris, but not Linux.
>
> They may true. If a difference cause a practical problem, please report it.

http://redmine.ruby-lang.org/issues/show/1169 - priority
http://redmine.ruby-lang.org/issues/show/2359 - loop{}

These are just two examples. Threading across platforms is very different. Heck, some platforms have multiple versions of threads.

> These two are wrong.
> Ruby 1.9 uses native threads but still use GIL just because of the problem you pointing out.

I know that the GIL is still being used, but this still doesn't protect data-structures that aren't ready for native threads. A C extension can still be thread-unsafe and fail in hard to understand and debug ways.

Having a new class that uses the native threads without the GIL everyplace would have been a better solution than potentially breaking.

Or using something that is completely data safe, like using erlang style message passing with processes, would be even better and more platform independent; C extensions wouldn't have to worry about being thread safe and locks wouldn't be needed for passing information around.
=end

#3 - 12/21/2009 03:22 AM - mame (Yusuke Endoh)

=begin
Hi,

2009/12/21 Christian Holtje redmine@ruby-lang.org:

> Issue #2505 has been updated by Christian Holtje.
>
>> Some examples:
>> priority doesn't do the same thing on Solaris than Linux.
>> loop{} will lock the process up in Solaris, but not Linux.
>
> They may true. If a difference cause a practical problem, please report it.
>
> http://redmine.ruby-lang.org/issues/show/1169 - priority

It is duplicated not only in Solaris but also in Linux. It is
difficult to support thread priority. IMO, we should think that
Thread#priority= is already deprecated.

> http://redmine.ruby-lang.org/issues/show/2359 - loop{}

Not duplicated.

> These are just two examples. Threading across platforms is very different. Heck, some platforms have multiple versions of threads.

You are right. Could you please report individually?

>> These two are wrong.
>> Ruby 1.9 uses native threads but still use GIL just because of the problem you pointing out.
>
> I know that the GIL is still being used, but this still doesn't protect data-structures that aren't ready for native threads. A C extension can still be thread-unsafe and fail in hard to understand and debug ways.

I cannot get your point. GIL locks not only code of Ruby but also
code of C extension. IWO, an execution of code in C extension is
not interrupted (unless you use BLOCKING_REGION or create new thread
explicitly). Even so, it can be thread-unsafe so easily?

> Having a new class that uses the native threads without the GIL everyplace would have been a better solution than potentially breaking.

It might be something that is currently developped, called MVM.

> Or using something that is completely data safe, like using erlang style message passing with processes, would be even better and more platform independent; C extensions wouldn't have to worry about being thread safe and locks wouldn't be needed for passing information around.

It is not Ruby :-(

--

Yusuke ENDOH mame@tsg.ne.jp

=end

#4 - 12/22/2009 09:39 PM - naruse (Yui NARUSE)

- Category set to core

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Priority changed from 5 to 3

=begin

=end

#5 - 04/17/2010 01:40 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Rejected

=begin
Hi,

I close this ticket.

> There are plenty of other examples.

Please register ticket for each.

> In addition, the fact that native threads use the same class name is also confusing.

Please answer my above question, and register Feature ticket
if you still want.

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