Ruby master - Misc #14901

[PATCH] do not block SIGCHLD in normal Ruby Threads

07/08/2018 02:53 AM - normalperson (Eric Wong)

Status:	Assigned	
Priority:	Normal	
Assignee:	normalperson (Eric Wong)	

Description

k0kubun (Takashi Kokubun): any opinions on this? Thanks.

```
I blocked SIGCHLD in normal Ruby Threads for [Bug #14867]
because I noticed at least two places which could not deal
with spurious wakeups in our test suite.

I also want to get rid of timer-thread due to resource
limitations [ruby-core:87773].  MJIT causes many SIGCHLD signals
so I found the following problems with cppflags=-DMJIT_FORCE_ENABLE=1

* OpenSSL::PKey::*.new does not resume on handle signals.
  rhenium acknowledged the problem and it should be in trunk soon:
  https://bugs.ruby-lang.org/issues/14882

* test/-ext-/gvl/test_last_thread.rb does not handle spurious
  wakeups.  Original report is in Japanese:
  https://bugs.ruby-lang.org/issues/11237
  I don't think it's a realistic expectation for code to be
  unable to deal with spurious wakeups.

One alternative could be to handle signals with MJIT thread
when MJIT is enabled, or to lazy-spawn timer thread to handle
signals when MJIT is enabled (MJIT + gcc requires a lot of
resources, anyways).
```

History

#1 - 07/08/2018 03:34 AM - k0kubun (Takashi Kokubun)

I have not completely read your patch for [Bug #14867] yet, so let me ask some questions to understand the context.

> I blocked SIGCHLD in normal Ruby Threads for [Bug #14867]

In the current trunk, in what kind of situation are normal Ruby Threads "blocked" by SIGCHLD? Are they blocked by default, or only during Process.waitall and its families are invoked?

And also, does the "blocked" mean interruption by a signal handler for SIGCHLD?

> MJIT causes many SIGCHLD signals

So whenever MJIT compiler process is spawned, normal Ruby Threads are blocked for now, right?

> One alternative could be to handle signals with MJIT thread when MJIT is enabled, or to lazy-spawn timer thread to handle signals when MJIT is enabled (MJIT + gcc requires a lot of resources, anyways).

If you're going to remove the Timer thread from normal Ruby execution, I'm in favor of handling signals with MJIT thread for simplicity, if it's not so hard to implement it in MJIT thread.

At a glance of your attached patch, it doesn't seem to implement either of it but to just give up the current approach. If tests with -DMJIT_FORCE_ENABLE are fine with the patch and problems in "OpenSSL::PKey::*.new" and " test/-ext-/gvl/test_last_thread.rb" are solved by it, go ahead to commit that. But if it's not the case, I wanna see your complete patch just in case.

#2 - 07/08/2018 07:04 AM - normalperson (Eric Wong)

takashikkbn@gmail.com wrote:

> I have not completely read your patch for [Bug #14867] yet, so let me ask some questions to understand the context.
>
> > I blocked SIGCHLD in normal Ruby Threads for [Bug #14867]
>
> In the current trunk, in what kind of situation are normal Ruby Threads "blocked" by SIGCHLD? Are they blocked by default, or only during Process.waitall and its families are invoked?
>
> And also, does the "blocked" mean interruption by a signal handler for SIGCHLD?

Ah, you seem to misunderstand, I suppose the terminology is confusing :x

In this context, "blocking" mean disabling interrupts using pthread_sigmask/sigprocmask. As in: blocking signals from being delivered to threads. Not blocking the threads themselves.

It is analogous to ec->interrupt_mask.

Before #14867: any thread can be interrupted by any signal at any time; aside from around fork/vfork/execve.

After #14897: any thread can receive non-SIGCHLD signals, only timer-thread sees SIGCHLD

This patch will restore pre-#14867 behavior.

My reasoning for this patch is that if any code is found broken by SIGCHLD from MJIT; it will ALSO (sooner or later) be found broken if hit by other signals. So the current disabling of SIGCHLD is just a hack to get tests to pass.

> If you're going to remove the Timer thread from normal Ruby execution, I'm in favor of handling signals with MJIT thread for simplicity, if it's not so hard to implement it in MJIT thread.

Right now, MJIT thread can already receive signals and run signal.c::sighandler (just like any other thread).
So maybe there's no need to do anything special, just let any thread handle any signals as before.

Another problem is mjit thread (or timer thread) doesn't always run, due to resource limitations or MJIT.pause, and we'd still need signal handling in those cases.

#3 - 07/08/2018 08:24 AM - k0kubun (Takashi Kokubun)

> In this context, "blocking" mean disabling interrupts using pthread_sigmask/sigprocmask. As in: blocking signals from being delivered to threads. Not blocking the threads themselves.

Oh I see. Thanks for the clarification.

> Before #14867: any thread can be interrupted by any signal at any time; aside from around fork/vfork/execve.
> After #14897: any thread can receive non-SIGCHLD signals, only timer-thread sees SIGCHLD
> This patch will restore pre-#14867 behavior.
> My reasoning for this patch is that if any code is found broken by SIGCHLD from MJIT; it will ALSO (sooner or later) be found broken if hit by other signals. So the current disabling of SIGCHLD is just a hack to get tests to pass.

So these 2 issues ("OpenSSL::PKey::*.new", test/-ext-/gvl/test_last_thread.rb) are (or should be) out of scope of [Bug #14867], which are still not resolved in a correct way, and this patch is going to remove the workaround to let them succeed. I can understand that direction.

If you already know they fail after committing the patch and they're going to be resolved separately, please skip them for now when RubyVM::MJIT.enabled?.

#4 - 07/08/2018 01:02 PM - k0kubun (Takashi Kokubun)

- Assignee changed from k0kubun (Takashi Kokubun) to normalperson (Eric Wong)

- Status changed from Open to Assigned

#5 - 07/08/2018 07:22 PM - normalperson (Eric Wong)

takashikkbn@gmail.com wrote:

> So these 2 issues ("OpenSSL::PKey::*.new",
> test/-ext-/gvl/test_last_thread.rb) are (or should be) out of
> scope of [Bug #14867], which are still not resolved in a
> correct way, and this patch is going to remove the workaround
> to let them succeed. I can understand that direction.
>
> If you already know they fail after committing the patch and
> they're going to be resolved separately, please skip them for
> now when RubyVM::MJIT.enabled?.

OK, thanks. I will probably use MJIT.enabled? test for
test_last_thread.rb, but OpenSSL::PKey* issue happens
in many places and I'll wait for rhenium to import ext/openssl

Files

0001-do-not-block-SIGCHLD-in-normal-Ruby-Threads.patch	2.5 KB	07/08/2018	normalperson (Eric Wong)