Ruby - Feature #5654

Introduce global lock to avoid concurrent require

11/21/2011 07:51 PM - nahi (Hiroshi Nakamura)

Status:	Assigned	
Priority:	Normal	
Assignee:	nahi (Hiroshi Nakamura)	
Target version:		

Description

=begin
Current implementation of "require" has locks for each file (expanded name from required feature) and serializes file loading from Threads. The first Thread acquires the lock for the file and starts loading. The second Thread waits for acquiring the lock, and when the first Thread release the lock, it acquires the lock, then returns immediately.
This can cause deadlock by cross-require from Threads.

And code that does not properly use "require" could meet several problems;

- constants can be defined before they're ready for use
- classes can be modified while they're being used
- global state can be initialized at the same time it's in use

Proposal: introduce global (per VM) lock to avoid concurrent file loading by "require" so that only one Thread can call "require" at the same time.

I don't have pros/cons list at this moment. Let's discuss it, too.

Derived from a discussion at #5621 (thread-safe autoload)
=end

Related issues:

Related to Ruby - Bug #11277: "code converter not found" error with multi-thr...	**Closed**

History

#1 - 11/22/2011 04:47 AM - wycats (Yehuda Katz)

The main caveat I can think of is that starting a reactor or other server loop inside a require will no longer work. I would argue that the benefits of much more deterministic require outweigh the costs of losing the ability to do this.

#2 - 11/29/2011 11:08 AM - spastorino (Santiago Pastorino)

As guys already said all the caveats I can think of are less important than allowing concurrent file loading

#3 - 12/06/2011 06:29 AM - Anonymous

> The main caveat I can think of is that starting a reactor or other server loop inside a require will no longer work. I would argue that the benefits of much more deterministic require outweigh the costs of losing the ability to do this.

+1
-r

#4 - 02/18/2012 02:27 AM - headius (Charles Nutter)

Finally looping back to this to add some thoughts. Sorry for the delay.

The biggest problem with concurrent code in Ruby (at least from the JRuby perspective) has not been thread safe collections, deadlocks, or anything in the core runtime...it has been the fact that lazy requires can cause a class to be visible before it is ready to be used.

Users are forced to choose between loading all files up front (and taking whatever startup hit comes along with that) or lazily loading files and hoping nobody tries to access the same class during that load. If requires were not allowed to run concurrently, there would be far less chance for a class to be partially initialized before it is used.

There are very few justifications for allowing concurrent requires.

One is the case Yehuda mentions, where you have a blocking call or loop in the body of a file you are requiring. I would argue this case is better served by loading, or by requiring the library you want to require and then calling a method. Using the example of the JVM...it is almost universally forbidden to do long, blocking operations in classes' static code executed on load, because it not only blocks loading but causes that thread to pause in the middle of initializing a given class.

The only other case would be to paralellize loading of many libraries. But this is flawed as well, since your loading process will potentially also see partially-initialized classes, and it could easily deadlock (with require locks as in 1.9) if libraries depend on each other.

So the only case that seems like it would be penalized is the case of running a server or loop or blocking call in the body of a script. If that case is deemed valid, perhaps "load" is the right choice instead. I do not believe "load" should have the same restriction as "require", since to me it represents the "raw" way to load code, and it can be an escape hatch for people who know what they're doing (like I hope those server/loop/blocking-call authors do).

A side note: the locking that has been added to "help" this problem has actually introduced additional problems. Specifically, both "thread safe autoload" and per-file require locks can lead to deadlocking at *load time*. This would be impossible if there were a single global lock for requires, and the goal of those fixes would still be fulfilled.

To further experimentation with this feature, we have added a flag to JRuby that turns on a global lock for requires: -Xglobal.require.lock=true. This is available in "master" builds of JRuby, downloadable from http://ci.jruby.org/snapshots/master.

I strongly encourage the Ruby elders to make requires lock on a single global lock.

#5 - 03/28/2012 12:34 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to nahi (Hiroshi Nakamura)

NaHi-san, could you facilitate the discussion?

--
Yusuke Endoh mame@tsg.ne.jp

#6 - 07/01/2012 12:24 AM - nahi (Hiroshi Nakamura)

- File 5654.pdf added

Endoh-san, here's "slide-show" of this proposal.

#7 - 07/01/2012 07:25 AM - nahi (Hiroshi Nakamura)

Please suggest how we can improve the slide here:
https://docs.google.com/presentation/d/1pXgUPtzoy4qFBznuuTSv0lzwmyoxUbxFyMk8GtzcXgI/edit

#8 - 07/02/2012 02:35 AM - mame (Yusuke Endoh)

NOT received because your slide includes no figure...!
Please make a presentation yourself at the meeting :-)

--
Yusuke Endoh mame@tsg.ne.jp

#9 - 07/02/2012 09:13 AM - nahi (Hiroshi Nakamura)

> NOT received

Fair because it's a regulation.

I updated the slide at https://docs.google.com/presentation/d/1pXgUPtzoy4qFBznuuTSv0lzwmyoxUbxFyMk8GtzcXgI/edit
Do you accept it now?

#10 - 07/02/2012 08:29 PM - mame (Yusuke Endoh)

Hello,

2012/7/2, nahi (Hiroshi Nakamura) nakahiro@gmail.com:

> > NOT received
>
> Fair because it's a regulation.

Sorry, just joking.

> I updated the slide at
> https://docs.google.com/presentation/d/1pXgUPtzoy4qFBznuuTSv0lzwmyoxUbxFyMk8GtzcXgI/edit
> Do you accept it now?

This project is for those who have no chance to appeal directly to matz.
But you have the chance. Actually, you will attend the meeting.
I think it is enough for you to make a presentation yourself.

--
Yusuke Endoh mame@tsg.ne.jp

#11 - 11/24/2012 08:35 AM - mame (Yusuke Endoh)

- Target version changed from 2.0.0 to 2.6

#12 - 06/19/2015 07:30 AM - ngoto (Naohisa Goto)

- Related to Bug #11277: "code converter not found" error with multi-thread (high occurrence rate since r50887) added

#13 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

Files

5654.pdf	34.1 KB	07/01/2012	nahi (Hiroshi Nakamura)