Ruby master - Bug #11384

multi-threaded autoload sometimes fails

07/22/2015 12:58 AM - normalperson (Eric Wong)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	trunk r51319	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: DONE, 2.4: DONE

Description

```
I get this failure once in a blue moon:

#8 test_autoload.rb:46:in `<top (required)>':
     open("zzz.rb", "w") {|f| f.puts "class ZZZ; def self.ok;:ok;end;end"}
     autoload :ZZZ, "./zzz.rb"
     t1 = Thread.new {ZZZ.ok}
     t2 = Thread.new {ZZZ.ok}
     [t1.value, t2.value].join
  #=> "" (expected "okok")
stderr output is not empty
   bootstraptest.tmp.rb:5:in `block in <main>': uninitialized constant ZZZ (Name
+Error)
test_autoload.rb        FAIL 1/8
FAIL 1/1010 tests failed


It is a very rare failure, I extracted it into a standalone script and
it took over 500,000 runs to hit it:

unless test(?e, "zzz.rb")
  open("zzz.rb", "w") {|f| f.puts "class ZZZ; def self.ok;:ok;end;end"}
end
autoload :ZZZ, "./zzz.rb"
t1 = Thread.new {ZZZ.ok}
t2 = Thread.new {ZZZ.ok}
[t1.value, t2.value].join
```

I'll work on this when I find time, but maybe somebody else can look at it
sooner. I'm not sure if it affects older versions.

Related issues:

Related to Ruby master - Bug #11683: multi-threaded autoload and defined? som...	**Closed**

Associated revisions

Revision cdc251c6 - 10/28/2015 11:59 PM - normal

variable.c: additional locking around autoload

[ruby-core:70075] [ruby-core:71239] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@52332 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 52332 - 10/28/2015 11:59 PM - normalperson (Eric Wong)

variable.c: additional locking around autoload

[ruby-core:70075] [ruby-core:71239] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

Revision 52332 - 10/28/2015 11:59 PM - normal

variable.c: additional locking around autoload

[ruby-core:70075] [ruby-core:71239] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

Revision 52332 - 10/28/2015 11:59 PM - normal

variable.c: additional locking around autoload

[ruby-core:70075] [ruby-core:71239] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

Revision 52332 - 10/28/2015 11:59 PM - normal

variable.c: additional locking around autoload

[ruby-core:70075] [ruby-core:71239] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

Revision 52332 - 10/28/2015 11:59 PM - normal

variable.c: additional locking around autoload

[ruby-core:70075] [ruby-core:71239] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

Revision 94a7a4e9 - 05/12/2017 09:52 PM - normal

autoload: always wait on loading thread

We cannot assume autoload_provided/rb_feature_provided returning
TRUE means it is safe to proceed without waiting. Another
thread may call rb_provide_feature before setting the constant
(via autoload_const_set). So we must wait until autoload is
completed by another thread.

Note: this patch was tested with an explicit rb_thread_schedule
in rb_provide_feature to make the race condition more apparent
as suggested by s.wanabe@gmail.com:

--- a/load.c
+++ b/load.c
@@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
rb_str_freeze(feature);

```
rb_ary_push(features, rb_fstring(feature));
```

+rb_thread_schedule();
features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
reset_loaded_features_snapshot();
}

- variable.c (check_autoload_required): do not assume a provided feature means autoload is complete, always wait if autoload is being performed by another thread. [ruby-core:81105] [Bug #11384] Thanks to s.wanabe@gmail.com

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58696 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 58696 - 05/12/2017 09:52 PM - normalperson (Eric Wong)

autoload: always wait on loading thread

We cannot assume autoload_provided/rb_feature_provided returning
TRUE means it is safe to proceed without waiting. Another
thread may call rb_provide_feature before setting the constant
(via autoload_const_set). So we must wait until autoload is
completed by another thread.

Note: this patch was tested with an explicit rb_thread_schedule
in rb_provide_feature to make the race condition more apparent
as suggested by s.wanabe@gmail.com:

--- a/load.c
+++ b/load.c
@@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
rb_str_freeze(feature);

```
rb_ary_push(features, rb_fstring(feature));
```

+rb_thread_schedule();
features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
reset_loaded_features_snapshot();
}

- variable.c (check_autoload_required): do not assume a provided feature means autoload is complete, always wait if autoload is being performed by another thread. [ruby-core:81105] [Bug #11384] Thanks to s.wanabe@gmail.com

Revision 58696 - 05/12/2017 09:52 PM - normal

autoload: always wait on loading thread

We cannot assume autoload_provided/rb_feature_provided returning
TRUE means it is safe to proceed without waiting. Another
thread may call rb_provide_feature before setting the constant
(via autoload_const_set). So we must wait until autoload is
completed by another thread.

Note: this patch was tested with an explicit rb_thread_schedule
in rb_provide_feature to make the race condition more apparent
as suggested by s.wanabe@gmail.com:

--- a/load.c
+++ b/load.c
@@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
rb_str_freeze(feature);

```
rb_ary_push(features, rb_fstring(feature));
```

+rb_thread_schedule();
features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
reset_loaded_features_snapshot();
}

- variable.c (check_autoload_required): do not assume a provided feature means autoload is complete, always wait if autoload is being performed by another thread. [ruby-core:81105] [Bug #11384] Thanks to s.wanabe@gmail.com

Revision 58696 - 05/12/2017 09:52 PM - normal

autoload: always wait on loading thread

We cannot assume autoload_provided/rb_feature_provided returning
TRUE means it is safe to proceed without waiting. Another
thread may call rb_provide_feature before setting the constant
(via autoload_const_set). So we must wait until autoload is
completed by another thread.

Note: this patch was tested with an explicit rb_thread_schedule
in rb_provide_feature to make the race condition more apparent
as suggested by s.wanabe@gmail.com:

```
--- a/load.c
+++ b/load.c
@@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
rb_str_freeze(feature);

 rb_ary_push(features, rb_fstring(feature));

+rb_thread_schedule();
features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
reset_loaded_features_snapshot();
}
```

- variable.c (check_autoload_required): do not assume a provided feature means autoload is complete, always wait if autoload is being performed by another thread. [ruby-core:81105] [Bug #11384] Thanks to s.wanabe@gmail.com

Revision e9ce3e74 - 06/30/2017 10:56 AM - usa (Usaku NAKAMURA)

merge revision(s) 58696: [Backport #11384]

```
    autoload: always wait on loading thread

    We cannot assume autoload_provided/rb_feature_provided returning
    TRUE means it is safe to proceed without waiting.  Another
    thread may call rb_provide_feature before setting the constant
    (via autoload_const_set).  So we must wait until autoload is
    completed by another thread.

    Note: this patch was tested with an explicit rb_thread_schedule
    in rb_provide_feature to make the race condition more apparent
    as suggested by <s.wanabe@gmail.com>:
    > --- a/load.c
    > +++ b/load.c
    > @@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
    >       rb_str_freeze(feature);
    >
    >       rb_ary_push(features, rb_fstring(feature));
    > +rb_thread_schedule();
    >       features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
    >       reset_loaded_features_snapshot();
    >  }

    * variable.c (check_autoload_required): do not assume a provided
      feature means autoload is complete, always wait if autoload is
      being performed by another thread.
      [ruby-core:81105] [Bug #11384] Thanks to <s.wanabe@gmail.com>
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_3@59221 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59221 - 06/30/2017 10:56 AM - usa (Usaku NAKAMURA)

merge revision(s) 58696: [Backport #11384]

```
autoload: always wait on loading thread

We cannot assume autoload_provided/rb_feature_provided returning
TRUE means it is safe to proceed without waiting.  Another
thread may call rb_provide_feature before setting the constant
(via autoload_const_set).  So we must wait until autoload is
completed by another thread.

Note: this patch was tested with an explicit rb_thread_schedule
in rb_provide_feature to make the race condition more apparent
as suggested by <s.wanabe@gmail.com>:
> --- a/load.c
```

```
> +++ b/load.c
> @@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
>       rb_str_freeze(feature);
>
>       rb_ary_push(features, rb_fstring(feature));
> +rb_thread_schedule();
>       features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
>       reset_loaded_features_snapshot();
>   }

* variable.c (check_autoload_required): do not assume a provided
  feature means autoload is complete, always wait if autoload is
  being performed by another thread.
  [ruby-core:81105] [Bug #11384] Thanks to <s.wanabe@gmail.com>
```

Revision 5e645629 - 07/09/2017 08:46 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 58696: [Backport #11384]

```
    autoload: always wait on loading thread

    We cannot assume autoload_provided/rb_feature_provided returning
    TRUE means it is safe to proceed without waiting.  Another
    thread may call rb_provide_feature before setting the constant
    (via autoload_const_set).  So we must wait until autoload is
    completed by another thread.

    Note: this patch was tested with an explicit rb_thread_schedule
    in rb_provide_feature to make the race condition more apparent
    as suggested by <s.wanabe@gmail.com>:
    > --- a/load.c
    > +++ b/load.c
    > @@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
    >       rb_str_freeze(feature);
    >
    >       rb_ary_push(features, rb_fstring(feature));
    > +rb_thread_schedule();
    >       features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
    >       reset_loaded_features_snapshot();
    >   }

    * variable.c (check_autoload_required): do not assume a provided
      feature means autoload is complete, always wait if autoload is
      being performed by another thread.
      [ruby-core:81105] [Bug #11384] Thanks to <s.wanabe@gmail.com>
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@59303 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59303 - 07/09/2017 08:46 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 58696: [Backport #11384]

```
autoload: always wait on loading thread

We cannot assume autoload_provided/rb_feature_provided returning
TRUE means it is safe to proceed without waiting.  Another
thread may call rb_provide_feature before setting the constant
(via autoload_const_set).  So we must wait until autoload is
completed by another thread.

Note: this patch was tested with an explicit rb_thread_schedule
in rb_provide_feature to make the race condition more apparent
as suggested by <s.wanabe@gmail.com>:
> --- a/load.c
> +++ b/load.c
> @@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
>       rb_str_freeze(feature);
>
>       rb_ary_push(features, rb_fstring(feature));
> +rb_thread_schedule();
>       features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
>       reset_loaded_features_snapshot();
>   }

* variable.c (check_autoload_required): do not assume a provided
```

```
feature means autoload is complete, always wait if autoload is
being performed by another thread.
[ruby-core:81105] [Bug #11384] Thanks to <s.wanabe@gmail.com>
```

History

#1 - 07/23/2015 10:28 PM - normalperson (Eric Wong)

Currently testing this in a loop:
http://80x24.org/spew/m/94541be0225540e34f0196e9754ae0eb5c07a4b7.txt

Subject: [PATCH] variable.c: additional locking around autoload

[ruby-core:70075] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

---

variable.c | 101 +++------------
1 file changed, 82 insertions(+), 19 deletions(-)

diff --git a/variable.c b/variable.c
index bb8b6db..1b6ddbd 100644
--- a/variable.c
+++ b/variable.c
@@ -16,6 +16,7 @@
#include "ruby/util.h"
#include "constant.h"
#include "id.h"
+#include "ccan/list/list.h"

st_table *rb_global_tbl;
static ID autoload, classpath, tmp_classpath, classid;
@@ -1880,6 +1881,7 @@ struct autoload_data_i {
int safe_level;
VALUE thread;
VALUE value;

- struct list_head waitq_head; };

static void
@@ -2060,20 +2062,70 @@ autoload_const_set(VALUE arg)
return 0; /* ignored */
}

+struct autoload_state {

- struct autoload_data_i *ele;
- VALUE mod;
- VALUE result;
- ID id;
- struct list_node waitq_node;
- VALUE waiting_th; +}; + static VALUE autoload_require(VALUE arg) {
- struct autoload_data_i *ele = (struct autoload_data_i *)arg;
- return rb_funcall(rb_vm_top_self(), rb_intern("require"), 1, ele->feature);
- struct autoload_state *state = (struct autoload_state *)arg; +
- /* this may release GVL and switch threads: */
- state->result = rb_funcall(rb_vm_top_self(), rb_intern("require"), 1,
- state->ele->feature); +
- return state->result; }

static VALUE
autoload_reset(VALUE arg)
{

- struct autoload_data_i *ele = (struct autoload_data_i *)arg;
- if (ele->thread == rb_thread_current()) {
- ele->thread = Qnil;
- struct autoload_state *state = (struct autoload_state *)arg;
- int need_wakeups = 0; +
- if (state->ele->thread == rb_thread_current()) {
- need_wakeups = 1;

- state->ele->thread = Qnil;
- } +
- /* At the last, move a value defined in autoload to constant table */
- if (RTEST(state->result) && state->ele->value != Qundef) {
- int safe_backup;
- struct autoload_const_set_args args; +
- args.mod = state->mod;
- args.id = state->id;
- args.value = state->ele->value;
- safe_backup = rb_safe_level();
- rb_set_safe_level_force(state->ele->safe_level);
- rb_ensure(autoload_const_set, (VALUE)&args,
- reset_safe, (VALUE)safe_backup);
- } +
- /* wakeup any waiters we had */
- if (need_wakeups) {
- struct autoload_state *cur, *nxt; +
- list_for_each_safe(&state->ele->waitq_head, cur, nxt, waitq_node) {
- VALUE th = cur->waiting_th; +
- cur->waiting_th = Qfalse;
- list_del(&cur->waitq_node); +
- /*
- * cur is stored on the stack of cur->waiting_th,
- * do not touch after waking up waiting_th
- */
- rb_thread_wakeup_alive(th);
- } } + return 0; /* ignored */ }

@@ -2083,6 +2135,7 @@ rb_autoload_load(VALUE mod, ID id)
VALUE load, result;
const char *loading = 0, *src;
struct autoload_data_i *ele;

- struct autoload_state state;

  if (!autoload_defined_p(mod, id)) return Qfalse;
  load = check_autoload_required(mod, id, &loading);
  @@ -2094,25 +2147,35 @@ rb_autoload_load(VALUE mod, ID id)
  if (!(ele = check_autoload_data(load))) {
  return Qfalse;
  }
  +

- state.ele = ele;

- state.mod = mod;

- state.id = id;
  if (ele->thread == Qnil) {
  ele->thread = rb_thread_current();
  +

- /*
  - autoload_reset will wake up any threads added to this
  - iff the GVL is released during autoload_require

- */

- list_head_init(&ele->waitq_head);
  }

- else {

- state.waiting_th = rb_thread_current();

- list_add_tail(&ele->waitq_head, &state.waitq_node);

- /*
  - autoload_reset in other thread will resume us and remove us
  - from the waitq list

- */
- do {
- [redacted]
- } while (state.waiting_th != Qfalse);
- }
  +
  /* autoload_data_i can be deleted by another thread while require */
- result = rb_ensure(autoload_require, (VALUE)ele, autoload_reset, (VALUE)ele);
- result = rb_ensure(autoload_require, (VALUE)&state,
- [redacted]
- if (RTEST(result)) {
- /* At the last, move a value defined in autoload to constant table */
- if (ele->value != Qundef) {
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- }
- }
  RB_GC_GUARD(load);
  return result;

  }

  EW

#2 - 07/24/2015 01:58 AM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> Currently testing this in a loop:
> http://80x24.org/spew/m/94541be0225540e34f0196e9754ae0eb5c07a4b7.txt

Nope, the original failure still happens with this, so there's
some other place where we're racing :<

#3 - 10/28/2015 05:54 AM - shugo (Shugo Maeda)

Eric Wong wrote:

> Eric Wong normalperson@yhbt.net wrote:
>
> > Currently testing this in a loop:
> > http://80x24.org/spew/m/94541be0225540e34f0196e9754ae0eb5c07a4b7.txt

> Nope, the original failure still happens with this, so there's
> some other place where we're racing :<

TestAutoload#test_threaded_accessing_constant has failed often since r52139,
and it seems to be the same problem.
I could reproduce the problem easily by the following script:

```
require "tempfile"

Tempfile.create(['autoload', '.rb']) {|file|
  file.puts 'sleep(0.5); class AutoloadTest; X = 1; end'
  file.close
  1.upto(Float::INFINITY) do |i|
    p i
    autoload(:AutoloadTest, file.path)
    begin
      t1 = Thread.new { ::AutoloadTest::X }
      t2 = Thread.new { ::AutoloadTest::X }
      [t1, t2].each(&:join)
    ensure
      if Object.const_defined?(:AutoloadTest)
        Object.send(:remove_const, :AutoloadTest)
        $".pop
      end
    end
  end
}
```

The script causes an error within 100 times on My Ubuntu 14.04 box,
but it runs over 10,000 times with your patch.

Your patch looks good to me, and at least it solves the race condition
of autoload_require(), even if there's still another race condition.
Why don't you commit it?

#4 - 10/28/2015 09:08 AM - normalperson (Eric Wong)

shugo@ruby-lang.org wrote:

> > Eric Wong normalperson@yhbt.net wrote:
> >
> > > http://80x24.org/spew/m/94541be0225540e34f0196e9754ae0eb5c07a4b7.txt
>
> Your patch looks good to me, and at least it solves the race condition
> of autoload_require(), even if there's still another race condition.
> Why don't you commit it?

Thanks for the reminder. I'll take a closer look at it tomorrow or day
after when I'm more awake. It's been eons since I wrote that
patch; maybe I did not fully understand the original code I replaced.

Will also take another look at Bug #10892, too.

#5 - 10/28/2015 11:18 PM - normalperson (Eric Wong)

shugo@ruby-lang.org wrote:

> The script causes an error within 100 times on My Ubuntu 14.04 box,
> but it runs over 10,000 times with your patch.

Is my patch still running beyond 10,000 times?

Without the patch, I can't reproduce the error within 200 times
(Debian wheezy + SMP).

#6 - 10/28/2015 11:58 PM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> shugo@ruby-lang.org wrote:

> > The script causes an error within 100 times on My Ubuntu 14.04 box,
> > but it runs over 10,000 times with your patch.
>
> Is my patch still running beyond 10,000 times?
>
> Without the patch, I can't reproduce the error within 200 times
> (Debian wheezy + SMP).

Nevermind, just took over 2000 times to reproduce on my system.
Will commit my patch + followups to reduce autoload_data_i size.

#7 - 10/28/2015 11:59 PM - Anonymous

- Status changed from Open to Closed

Applied in changeset r52332.

---

variable.c: additional locking around autoload

[ruby-core:70075] [ruby-core:71239] [Bug #11384]

Note: this open-coding locking method may go into
rb_mutex/rb_thread_shield types. It is smaller and simpler and
based on the wait queue implementation of the Linux kernel.

When/if we get rid of GVL, native mutexes may be used as-is.

#8 - 10/29/2015 02:09 AM - shugo (Shugo Maeda)

Eric Wong wrote:

> shugo@ruby-lang.org wrote:
>
> > The script causes an error within 100 times on My Ubuntu 14.04 box,
> > but it runs over 10,000 times with your patch.
>
> Is my patch still running beyond 10,000 times?

It's still running beyond 49,000 times on my notebook, which sleeps at night.
I'll inform you if any error occurs.

#9 - 11/03/2015 03:38 AM - normalperson (Eric Wong)

shugo@ruby-lang.org wrote:

> It's still running beyond 49,000 times on my notebook, which sleeps at night.
> I'll inform you if any error occurs.

Actually, I've managed to reproduce around 47,000 times.
So my patch (r52332) seems to help make the problem less obvious,
but yeah, there's still a bug somewhere...

#10 - 05/11/2017 11:43 PM - wanabe (_ wanabe)

- Status changed from Closed to Open

I re-open this ticket because it remains the issue.
Please do not hesitate to close and open new one if you want.

I think this is an issue of the timing between rb_provide_feature() and autoload_const_set().

This is reproduction code that is to reference to https://bugs.ruby-lang.org/issues/11384#note-3 and test_autoload.rb.

```
require "tempfile"

Tempfile.create(['autoload', '.rb']) {|file|
  file.puts 'ZZZ = 1'
  file.close
```

```
  1.upto(Float::INFINITY) do |i|
    STDERR.print "#{i}\n"
    autoload(:ZZZ, file.path)
    begin
      t1 = Thread.new { ZZZ }
      t2 = Thread.new { Thread.pass; ZZZ }
      Thread.pass
      [t1, t2].each(&:join)
    ensure
      if Object.const_defined?(:ZZZ)
        Object.send(:remove_const, :ZZZ)
        $".pop
      end
    end
  end
}
```

This is debug print patch.

```
diff --git a/load.c b/load.c
index 75ac4df83f..58dbb47382 100644
--- a/load.c
+++ b/load.c
@@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
     rb_str_freeze(feature);

     rb_ary_push(features, rb_fstring(feature));
+fprintf(stderr, "%p >>> rb_provide_feature -> rb_ary_push\n", rb_thread_current());
     features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
     reset_loaded_features_snapshot();
 }
diff --git a/variable.c b/variable.c
index 6e883c7041..73d040dd61 100644
--- a/variable.c
+++ b/variable.c
@@ -2085,6 +2085,7 @@ autoload_reset(VALUE arg)
    rb_set_safe_level_force(state->ele->safe_level);
    rb_ensure(autoload_const_set, (VALUE)&args,
              reset_safe, (VALUE)safe_backup);
+fprintf(stderr, "%p <<< autoload_reset <- autoload_const_set\n", rb_thread_current());
     }

     /* wakeup any waiters we had */
@@ -2144,8 +2145,11 @@ rb_autoload_load(VALUE mod, ID id)
     struct autoload_data_i *ele;
     struct autoload_state state;

+fprintf(stderr, "%p check_autoload_required 0\n", rb_thread_current());
     if (!autoload_defined_p(mod, id)) return Qfalse;
+fprintf(stderr, "%p check_autoload_required 1\n", rb_thread_current());
     load = check_autoload_required(mod, id, &loading);
+fprintf(stderr, "%p check_autoload_required 2 %p\n", rb_thread_current(), load);
     if (!load) return Qfalse;
     src = rb_sourcefile();
     if (src && loading && strcmp(src, loading) == 0) return Qfalse;
```

This is a short extract from output.
(Sorry, I can't attached entire log because too large: 3.2 MB > 2 MB)

```
12170
0x7f0004007340 check_autoload_required 0
0x7f0004007340 check_autoload_required 1
0x7f0004007340 check_autoload_required 2 0x7f0004007390
0x7f0004007340 >>> rb_provide_feature -> rb_ary_push
0x7f0004007228 check_autoload_required 0
0x7f0004007228 check_autoload_required 1
0x7f0004007228 check_autoload_required 2 (nil)
0x7f0004007340 <<< autoload_reset <- autoload_const_set
a.rb:11:in `block (3 levels) in ': uninitialized constant ZZZ (NameError)
```

When thread 0x7f0004007228 called check_autoload_required(),
thread 0x7f0004007340 had pushed load script path into $" but had not yet called autoload_const_set().

If you want to reproduce easily, rb_thread_schedule() may help you.

```
diff --git a/load.c b/load.c
index 75ac4df83f..2d4172e112 100644
--- a/load.c
+++ b/load.c
@@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
     rb_str_freeze(feature);

     rb_ary_push(features, rb_fstring(feature));
+rb_thread_schedule();
     features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
     reset_loaded_features_snapshot();
 }
```

#11 - 05/12/2017 12:08 AM - normalperson (Eric Wong)

s.wanabe@gmail.com wrote:

> Status changed from Closed to Open
>
> I re-open this ticket because it remains the issue.
> Please do not hesitate to close and open new one if you want.

No worries, I don't care for ticket organization; all that
matters is problems get fixed. I will investigate this later
today, or tomorrow at latest. Thank you for reporting the
problem.

#12 - 05/12/2017 09:51 PM - normalperson (Eric Wong)

- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN to 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: REQUIRED, 2.4: REQUIRED

- File 0001-autoload-always-wait-on-loading-thread.patch added

I think the attached patch should fix it, over 12 million iterations
and still going strong.

#13 - 05/12/2017 09:52 PM - Anonymous

- Status changed from Open to Closed

Applied in changeset trunk|r58696.

---

autoload: always wait on loading thread

We cannot assume autoload_provided/rb_feature_provided returning
TRUE means it is safe to proceed without waiting. Another
thread may call rb_provide_feature before setting the constant
(via autoload_const_set). So we must wait until autoload is
completed by another thread.

Note: this patch was tested with an explicit rb_thread_schedule
in rb_provide_feature to make the race condition more apparent
as suggested by s.wanabe@gmail.com:

> --- a/load.c
> +++ b/load.c
> @@ -563,6 +563,7 @@ rb_provide_feature(VALUE feature)
> rb_str_freeze(feature);
>
> rb_ary_push(features, rb_fstring(feature));
>
> +rb_thread_schedule();
> features_index_add(feature, INT2FIX(RARRAY_LEN(features)-1));
> reset_loaded_features_snapshot();
> }

- variable.c (check_autoload_required): do not assume a provided feature means autoload is complete, always wait if autoload is being performed by another thread. [ruby-core:81105] [Bug #11384] Thanks to s.wanabe@gmail.com

#14 - 06/30/2017 10:56 AM - usa (Usaku NAKAMURA)

- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: REQUIRED, 2.4: REQUIRED to 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: DONE, 2.4: REQUIRED

ruby_2_3 r59221 merged revision(s) 58696.

#15 - 07/09/2017 08:46 PM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: DONE, 2.4: REQUIRED to 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: DONE, 2.4: DONE

ruby_2_4 r59303 merged revision(s) 58696.

#16 - 07/23/2019 09:14 PM - jeremyevans0 (Jeremy Evans)

- Related to Bug #11683: multi-threaded autoload and defined? sometimes fails added

Files

0001-autoload-always-wait-on-loading-thread.patch	1.96 KB	05/12/2017	normalperson (Eric Wong)