Ruby master - Bug #13150

TestMarshal failures on FreeBSD with gcc7 because of GC

01/23/2017 04:58 PM - naruse (Yui NARUSE)

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

Description

```
  1) Failure:
TestMarshal#test_context_switch [/home/naruse/ruby/test/ruby/test_marshal.rb:368]:
[StopIteration] exception expected, not.
Class: <RuntimeError>
Message: <"Marshal.dump reentered at marshal_dump">
---Backtrace---
/home/naruse/ruby/test/ruby/test_marshal.rb:345:in `dump'
/home/naruse/ruby/test/ruby/test_marshal.rb:345:in `dump_each'
../../ruby/test/runner.rb: TestMarshal#test_context_switch:1:in `each'
---------------

  2) Failure:
TestMarshal#test_gc [/home/naruse/ruby/test/ruby/test_marshal.rb:187]:
Exception raised:
<#<RuntimeError: Marshal.dump reentered at _dump>>.

Finished tests in 0.153251s, 698.2016 tests/s, 6238.1380 assertions/s.
107 tests, 956 assertions, 2 failures, 0 errors, 0 skips

ruby -v: ruby 2.5.0dev (2017-01-23 trunk 57407) [x86_64-freebsd10.3]
```

Related issues:

Has duplicate Ruby master - Bug #13168: Marshaling broken with GCC 7.x	**Closed**
Has duplicate Ruby master - Bug #13319: GC issues seen with GCC7	**Closed**

Associated revisions

Revision 7c1b30a6 - 01/23/2017 04:58 PM - naruse (Yui NARUSE)

Prevent GC by volatile [Bug #13150]

test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
20170115 (experimental); RB_GC_GUARD looks not worked well.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57410 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57410 - 01/23/2017 04:58 PM - naruse (Yui NARUSE)

Prevent GC by volatile [Bug #13150]

test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
20170115 (experimental); RB_GC_GUARD looks not worked well.

Revision 57410 - 01/23/2017 04:58 PM - naruse (Yui NARUSE)

Prevent GC by volatile [Bug #13150]

test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
20170115 (experimental); RB_GC_GUARD looks not worked well.

Revision 57410 - 01/23/2017 04:58 PM - naruse (Yui NARUSE)

Prevent GC by volatile [Bug #13150]

test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
20170115 (experimental); RB_GC_GUARD looks not worked well.

Revision 8788489c - 03/13/2017 07:30 AM - naruse (Yui NARUSE)

merge revision(s) 57410,57619,57621,57631,57634: [Backport #13150]

```
    Prevent GC by volatile [Bug #13150]

    test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
    are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
    20170115 (experimental); RB_GC_GUARD looks not worked well.
    ruby.h: RB_GC_GUARD stronger than gcc7

    * include/ruby/ruby.h (RB_GC_GUARD): prevent guarded pointer from
      optimization by using as an input to inline asm.
    ruby.h: remove comment

    * include/ruby/ruby.h (RB_GC_GUARD): remove comment unsupported by
      Solaris AS.
    marshal.c: use hidden objects to allow recycling

    Hidden objects (klass == 0) are not visible to Ruby code invoked
    from other threads or signal handlers, so they can never be
    accessed from other contexts.  This makes it safe to call
    rb_gc_force_recycle on the object slot after releasing malloc
    memory.

    * marshal.c (rb_marshal_dump_limited): hide dump_arg and recycle when done
      (rb_marshal_load_with_proc): hide load_arg and recycle when done
      [ruby-core:79518]
    marshal.c: revert r57631 partially

    * marshal.c (rb_marshal_dump_limited): do not free dump_arg, which
      may be dereferenced in check_dump_arg due to continuation, and
      get rid of dangling pointers.

    * marshal.c (rb_marshal_load_with_proc): ditto for load_arg.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@57954 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57954 - 03/13/2017 07:30 AM - naruse (Yui NARUSE)

merge revision(s) 57410,57619,57621,57631,57634: [Backport #13150]

```
Prevent GC by volatile [Bug #13150]

test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
20170115 (experimental); RB_GC_GUARD looks not worked well.
ruby.h: RB_GC_GUARD stronger than gcc7

* include/ruby/ruby.h (RB_GC_GUARD): prevent guarded pointer from
  optimization by using as an input to inline asm.
ruby.h: remove comment

* include/ruby/ruby.h (RB_GC_GUARD): remove comment unsupported by
  Solaris AS.
marshal.c: use hidden objects to allow recycling

Hidden objects (klass == 0) are not visible to Ruby code invoked
from other threads or signal handlers, so they can never be
accessed from other contexts.  This makes it safe to call
rb_gc_force_recycle on the object slot after releasing malloc
memory.

* marshal.c (rb_marshal_dump_limited): hide dump_arg and recycle when done
  (rb_marshal_load_with_proc): hide load_arg and recycle when done
  [ruby-core:79518]
marshal.c: revert r57631 partially

* marshal.c (rb_marshal_dump_limited): do not free dump_arg, which
  may be dereferenced in check_dump_arg due to continuation, and
  get rid of dangling pointers.
```

```
* marshal.c (rb_marshal_load_with_proc): ditto for load_arg.
```

Revision c3205d65 - 08/09/2017 08:40 AM - usa (Usaku NAKAMURA)

[Backport #13150]

```
    this patch contains r54158, r57410, r57631 and r57954.

    Prevent GC by volatile [Bug #13150]

    test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
    are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
    20170115 (experimental); RB_GC_GUARD looks not worked well.

    * include/ruby/ruby.h (RB_GC_GUARD): prevent guarded pointer from
      optimization by using as an input to inline asm.

    * ruby.h: remove comment

    * include/ruby/ruby.h (RB_GC_GUARD): remove comment unsupported by
      Solaris AS.

    Hidden objects (klass == 0) are not visible to Ruby code invoked
    from other threads or signal handlers, so they can never be
    accessed from other contexts.  This makes it safe to call
    rb_gc_force_recycle on the object slot after releasing malloc
    memory.

    * marshal.c (rb_marshal_dump_limited): hide dump_arg and recycle when
      done (rb_marshal_load_with_proc): hide load_arg and recycle when done
      [ruby-core:79518]

    * marshal.c (rb_marshal_dump_limited): do not free dump_arg, which
      may be dereferenced in check_dump_arg due to continuation, and
      get rid of dangling pointers.

    * marshal.c (rb_marshal_load_with_proc): ditto for load_arg.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_3@59539 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59539 - 08/09/2017 08:40 AM - usa (Usaku NAKAMURA)

[Backport #13150]

```
this patch contains r54158, r57410, r57631 and r57954.

Prevent GC by volatile [Bug #13150]

test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
20170115 (experimental); RB_GC_GUARD looks not worked well.

* include/ruby/ruby.h (RB_GC_GUARD): prevent guarded pointer from
  optimization by using as an input to inline asm.

* ruby.h: remove comment

* include/ruby/ruby.h (RB_GC_GUARD): remove comment unsupported by
  Solaris AS.

Hidden objects (klass == 0) are not visible to Ruby code invoked
from other threads or signal handlers, so they can never be
accessed from other contexts.  This makes it safe to call
rb_gc_force_recycle on the object slot after releasing malloc
memory.

* marshal.c (rb_marshal_dump_limited): hide dump_arg and recycle when
  done (rb_marshal_load_with_proc): hide load_arg and recycle when done
  [ruby-core:79518]

* marshal.c (rb_marshal_dump_limited): do not free dump_arg, which
  may be dereferenced in check_dump_arg due to continuation, and
  get rid of dangling pointers.
```

```
* marshal.c (rb_marshal_load_with_proc): ditto for load_arg.
```

Revision ecc889b1 - 08/09/2017 12:17 PM - usa (Usaku NAKAMURA)

merge revision(s) 57634: [Backport #13150]

```
    marshal.c: revert r57631 partially

    * marshal.c (rb_marshal_dump_limited): do not free dump_arg, which
      may be dereferenced in check_dump_arg due to continuation, and
      get rid of dangling pointers.

    * marshal.c (rb_marshal_load_with_proc): ditto for load_arg.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_3@59551 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59551 - 08/09/2017 12:17 PM - usa (Usaku NAKAMURA)

merge revision(s) 57634: [Backport #13150]

```
marshal.c: revert r57631 partially

* marshal.c (rb_marshal_dump_limited): do not free dump_arg, which
  may be dereferenced in check_dump_arg due to continuation, and
  get rid of dangling pointers.

* marshal.c (rb_marshal_load_with_proc): ditto for load_arg.
```

History

#1 - 01/23/2017 04:58 PM - naruse (Yui NARUSE)

- Status changed from Open to Closed

Applied in changeset r57410.

Prevent GC by volatile [Bug #13150]

test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
20170115 (experimental); RB_GC_GUARD looks not worked well.

#2 - 01/23/2017 07:41 PM - normalperson (Eric Wong)

naruse@ruby-lang.org wrote:

```
    Prevent GC by volatile [Bug #13150]

    test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
    are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
    20170115 (experimental); RB_GC_GUARD looks not worked well.
```

Have you tried to experiment with making RB_GC_GUARD stronger
for gcc7, instead? There may be similar bugs, and I would rather
improve RB_GC_GUARD than add volatiles (which are more subtle
and have more variance between implementations).

Thanks

> Modified files:
> trunk/marshal.c

#3 - 01/31/2017 01:41 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Bug #13168: Marshaling broken with GCC 7.x added

#4 - 01/31/2017 01:42 AM - nobu (Nobuyoshi Nakada)

- Description updated

#5 - 02/03/2017 02:31 PM - vo.x (Vit Ondruch)

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

It would be nice to backport this, so we don't need to carry the patch around in Fedora.

#6 - 02/13/2017 01:08 AM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> naruse@ruby-lang.org wrote:
>
> ```
> Prevent GC by volatile [Bug #13150]
>
> test/ruby/test_marshal.rb test_context_switch (load) and test_gc (dump)
> are failed on FreeBSD 10.3 and gcc7 (FreeBSD Ports Collection) 7.0.0
> 20170115 (experimental); `RB_GC_GUARD` looks not worked well.
> ```
>
> Have you tried to experiment with making RB_GC_GUARD stronger
> for gcc7, instead? There may be similar bugs, and I would rather
> improve RB_GC_GUARD than add volatiles (which are more subtle
> and have more variance between implementations).

I still support fixing RB_GC_GUARD to be stronger for GCC7.

But in marshal.c, I think we can use klass==0 to hide the object
and use rb_gc_force_recycle, instead. AFAIK,
rb_gc_force_recycle is safe if the object has klass==0 for its
entire lifetime.

How about the following?

```
diff --git a/marshal.c b/marshal.c
index d628daa4de..b9c9e6a03e 100644
--- a/marshal.c
+++ b/marshal.c
@@ -1024,9 +1024,9 @@ VALUE
 rb_marshal_dump_limited(VALUE obj, VALUE port, int limit)
 {
     struct dump_arg *arg;
-    volatile VALUE wrapper; /* used to avoid memory leak in case of exception */
+    VALUE wrapper; /* used to avoid memory leak in case of exception */

-    wrapper = TypedData_Make_Struct(rb_cData, struct dump_arg, &dump_arg_data, arg);
+    wrapper = TypedData_Make_Struct(0, struct dump_arg, &dump_arg_data, arg);
     arg->dest = 0;
     arg->symbols = st_init_numtable();
     arg->data    = rb_init_identtable();
@@ -1053,8 +1053,8 @@ rb_marshal_dump_limited(VALUE obj, VALUE port, int limit)
     rb_io_write(arg->dest, arg->str);
     rb_str_resize(arg->str, 0);
     }
-    clear_dump_arg(arg);
-    RB_GC_GUARD(wrapper);
+    free_dump_arg(arg);
+    rb_gc_force_recycle(wrapper);

     return port;
 }
@@ -2038,7 +2038,7 @@ rb_marshal_load_with_proc(VALUE port, VALUE proc)
 {
     int major, minor, infection = 0;
     VALUE v;
-    volatile VALUE wrapper; /* used to avoid memory leak in case of exception */
+    VALUE wrapper; /* used to avoid memory leak in case of exception */
     struct load_arg *arg;

     v = rb_check_string_type(port);
@@ -2053,7 +2053,7 @@ rb_marshal_load_with_proc(VALUE port, VALUE proc)
     else {
     io_needed();
     }
-    wrapper = TypedData_Make_Struct(rb_cData, struct load_arg, &load_arg_data, arg);
+    wrapper = TypedData_Make_Struct(0, struct load_arg, &load_arg_data, arg);
     arg->infection = infection;
```

```
     arg->src = port;
     arg->offset = 0;
@@ -2084,8 +2084,8 @@ rb_marshal_load_with_proc(VALUE port, VALUE proc)

     if (!NIL_P(proc)) arg->proc = proc;
     v = r_object(arg);
-    clear_load_arg(arg);
-    RB_GC_GUARD(wrapper);
+    free_load_arg(arg);
+    rb_gc_force_recycle(wrapper);

     return v;
 }
```

#7 - 02/13/2017 10:08 AM - nobu (Nobuyoshi Nakada)

On 2017/02/13 10:04, Eric Wong wrote:

> I still support fixing RB_GC_GUARD to be stronger for GCC7.

I think it is stronger than GCC7 now.

> But in marshal.c, I think we can use klass==0 to hide the object
> and use rb_gc_force_recycle, instead. AFAIK,
> rb_gc_force_recycle is safe if the object has klass==0 for its
> entire lifetime.
>
> How about the following?

Seems nice.

--
Nobu Nakada

#8 - 02/15/2017 07:32 AM - nobu (Nobuyoshi Nakada)

On 2017/02/13 19:05, Nobuyoshi Nakada wrote:

> > But in marshal.c, I think we can use klass==0 to hide the object
> > and use rb_gc_force_recycle, instead. AFAIK,
> > rb_gc_force_recycle is safe if the object has klass==0 for its
> > entire lifetime.
> >
> > How about the following?
>
> Seems nice.

Sorry, I missed that arg may be dereferenced in check_dump_arg()
in the case continuation is used. Hiding wrapper objects is fine, but
freeing arg and recycling wrapper causes a dangling pointer and
can segfault on some environments, compilers and options, with the
following pacth.

```
diff --git a/test/ruby/test_marshal.rb b/test/ruby/test_marshal.rb
index bc22b5fd3a..bfc3f6df25 100644
--- a/test/ruby/test_marshal.rb
+++ b/test/ruby/test_marshal.rb
@@ -644,6 +644,9 @@
       c = Bug9523.new
       assert_raise_with_message(RuntimeError, /Marshal\.dump reentered at marshal_dump/) do
       Marshal.dump(c)
+      GC.start
+      1000.times {"x"*1000}
+      GC.start
       c.cc.call
     end
   end
```

#9 - 02/15/2017 08:51 AM - normalperson (Eric Wong)

Nobuyoshi Nakada nobu@ruby-lang.org wrote:

> On 2017/02/13 19:05, Nobuyoshi Nakada wrote:
>
> > But in marshal.c, I think we can use klass==0 to hide the object
> > and use rb_gc_force_recycle, instead. AFAIK,
> > rb_gc_force_recycle is safe if the object has klass==0 for its
> > entire lifetime.
> >
> > How about the following?
>
> Seems nice.

> Sorry, I missed that arg may be dereferenced in check_dump_arg()
> in the case continuation is used. Hiding wrapper objects is fine, but
> freeing arg and recycling wrapper causes a dangling pointer and
> can segfault on some environments, compilers and options, with the
> following pacth.

Ah, thanks. I forgot this :x I saw you already made r57634.
Since callcc is obsolete, is the following patch OK; or can Fiber
have the same problem?

(Admittedly, I never fully understood callcc :x)

```
diff --git a/cont.c b/cont.c
index 50fa45e96e..363e05fb91 100644
--- a/cont.c
+++ b/cont.c
@@ -149,7 +149,7 @@ struct rb_fiber_struct {
 };

 static const rb_data_type_t cont_data_type, fiber_data_type;
-static VALUE rb_cContinuation;
+VALUE rb_cContinuation;
 static VALUE rb_cFiber;
 static VALUE rb_eFiberError;

diff --git a/marshal.c b/marshal.c
index 2a10b98100..57acf8c30c 100644
--- a/marshal.c
+++ b/marshal.c
@@ -20,6 +20,8 @@
 #include "encindex.h"
 #include "id_table.h"

+extern VALUE rb_cContinuation;
+
 #include <math.h>
 #ifdef HAVE_FLOAT_H
 #include <float.h>
@@ -1053,8 +1055,14 @@ rb_marshal_dump_limited(VALUE obj, VALUE port, int limit)
     rb_io_write(arg->dest, arg->str);
     rb_str_resize(arg->str, 0);
     }
-    clear_dump_arg(arg);
-    RB_GC_GUARD(wrapper);
+    if (rb_cContinuation) {
+   clear_dump_arg(arg);
+   RB_GC_GUARD(wrapper);
+    }
+    else {
+   free_dump_arg(arg);
+   rb_gc_force_recycle(wrapper);
+    }

     return port;
 }
@@ -2084,8 +2092,14 @@ rb_marshal_load_with_proc(VALUE port, VALUE proc)

     if (!NIL_P(proc)) arg->proc = proc;
     v = r_object(arg);
-    free_load_arg(arg);
-    rb_gc_force_recycle(wrapper);
```

```
+    if (rb_cContinuation) {
+   clear_load_arg(arg);
+   RB_GC_GUARD(wrapper);
+    }
+    else {
+   free_load_arg(arg);
+   rb_gc_force_recycle(wrapper);
+    }

     return v;
 }
```

#10 - 03/13/2017 07:30 AM - naruse (Yui NARUSE)

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

ruby_2_4 r57954 merged revision(s) 57410,57619,57621,57631,57634.

#11 - 03/14/2017 03:40 AM - nagachika (Tomoyuki Chikanaga)

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

#12 - 03/14/2017 01:35 PM - nobu (Nobuyoshi Nakada)

normalperson (Eric Wong) wrote:

> Ah, thanks. I forgot this :x I saw you already made r57634.
> Since callcc is obsolete, is the following patch OK; or can Fiber
> have the same problem?

I guess Fiber would too.

#13 - 04/19/2017 02:20 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Bug #13319: GC issues seen with GCC7 added

#14 - 07/19/2017 03:04 AM - hsbt (Hiroshi SHIBATA)

- File ruby_2_3_gcc7.patch added

> usa

I got a report for this issue about ruby-build repository.

https://github.com/rbenv/ruby-build/issues/1115

I created a patch for ruby_2_3 branch with gcc7. this patch contains r54158, r57410, r57631 and r57954.

#15 - 07/24/2017 03:14 AM - MSP-Greg (Greg L)

Hiroshi,

Thank you for the patch. I just decided to start doing regular MinGW builds with ruby_2_3 & ruby_2_4. I also recently updated to gcc 7.1.0 from 6.3.0. With 7.1.0, ruby_2_4 was fine, but ruby_2_3 didn't get very far. Your patch seems to have solved the issue with MinGW & 7.1.0. Below are initial test results -

ruby 2.3.5p342 (2017-07-07 revision 59277) [x64-mingw32]
test-btest passed
test-all 15487 tests, 2192347 assertions, 4 failures, 1 errors, 216 skips

Haven't hooked up spec tests yet...

#16 - 08/09/2017 08:41 AM - usa (Usaku NAKAMURA)

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

#17 - 02/27/2018 09:27 PM - Eregon (Benoit Daloze)

This still happens on 2.2.9, should it be backported too?
(On gcc (GCC) 7.2.1 20170915 (Red Hat 7.2.1-2))

Files

ruby_2_3_gcc7.patch	3.1 KB	07/19/2017	hsbt (Hiroshi SHIBATA)