Ruby - Bug #2777

Invalid read of size 4 by redefining load

02/22/2010 03:03 PM - nobu (Nobuyoshi Nakada)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	1.9.2		
ruby -v:	-	**Backport:**	

Description

=begin
なかだです。

At Mon, 22 Feb 2010 01:20:07 +0900,
Tanaka Akira wrote in [ruby-dev:40452]:

> 以下のように load 中に load を再定義すると、変なところをアクセスするのが
> valgrind で観測されます。

rb_method_entry_tにもrefcountを入れますかねぇ。

--
--- 僕の前にBugはない。
--- 僕の後ろにBugはできる。
中田 伸悦
=end

Associated revisions

Revision 833cade2dce8ee8a9dd2091fcc84880030a51d54 - 05/05/2010 05:51 PM - ko1 (Koichi Sasada)

- vm_method.c (rb_unlink_method_entry, rb_sweep_method_entry):
  added. Unlinked method entries are collected to
  vm->unlinked_method_entry_list. On the GC timing, mark all method
  entries which are on all living threads. Only non-marked method
  entries are collected. This hack prevents releasing living method
  entry.
  [Performance Consideration] Since this Method Entry GC (MEGC)
  doesn't occuer frequently, MEGC will not be a performance bottleneck.
  However, to traverse living method entries, every control frame push
  needs to clear cfp->me field. This will be a performance issue
  (because pushing control frame is occurred frequently).
  Bug #2777 [ruby-dev:40457]
- cont.c (fiber_init): init cfp->me.
- gc.c (garbage_collect): kick rb_sweep_method_entry().
- method.h (rb_method_entry_t): add a mark field.
- vm.c (invoke_block_from_c): set passed me.
- vm.c (rb_thread_mark): mark cfp->me.
- vm_core.h (rb_thread_t): add a field passed_me.
- vm_core.h (rb_vm_t): add a field unlinked_method_entry_list.
- vm_insnhelper.c (vm_push_frame): clear cfp->me at all times.
- vm_insnhelper.c (vm_call_bmethod): pass me.
- bootstraptest/test_method.rb: add a test.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@27634 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 833cade2dce8ee8a9dd2091fcc84880030a51d54 - 05/05/2010 05:51 PM - ko1 (Koichi Sasada)

- vm_method.c (rb_unlink_method_entry, rb_sweep_method_entry):
  added. Unlinked method entries are collected to
  vm->unlinked_method_entry_list. On the GC timing, mark all method
  entries which are on all living threads. Only non-marked method
  entries are collected. This hack prevents releasing living method
  entry.

[Performance Consideration] Since this Method Entry GC (MEGC) doesn't occuer frequently, MEGC will not be a performance bottleneck. However, to traverse living method entries, every control frame push needs to clear cfp->me field. This will be a performance issue (because pushing control frame is occurred frequently).
Bug #2777 [ruby-dev:40457]

- cont.c (fiber_init): init cfp->me.
- gc.c (garbage_collect): kick rb_sweep_method_entry().
- method.h (rb_method_entry_t): add a mark field.
- vm.c (invoke_block_from_c): set passed me.
- vm.c (rb_thread_mark): mark cfp->me.
- vm_core.h (rb_thread_t): add a field passed_me.
- vm_core.h (rb_vm_t): add a field unlinked_method_entry_list.
- vm_insnhelper.c (vm_push_frame): clear cfp->me at all times.
- vm_insnhelper.c (vm_call_bmethod): pass me.
- bootstraptest/test_method.rb: add a test.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@27634 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 833cade2 - 05/05/2010 05:51 PM - ko1 (Koichi Sasada)

- vm_method.c (rb_unlink_method_entry, rb_sweep_method_entry): added. Unlinked method entries are collected to vm->unlinked_method_entry_list. On the GC timing, mark all method entries which are on all living threads. Only non-marked method entries are collected. This hack prevents releasing living method entry.
  [Performance Consideration] Since this Method Entry GC (MEGC) doesn't occuer frequently, MEGC will not be a performance bottleneck. However, to traverse living method entries, every control frame push needs to clear cfp->me field. This will be a performance issue (because pushing control frame is occurred frequently).
  Bug #2777 [ruby-dev:40457]
- cont.c (fiber_init): init cfp->me.
- gc.c (garbage_collect): kick rb_sweep_method_entry().
- method.h (rb_method_entry_t): add a mark field.
- vm.c (invoke_block_from_c): set passed me.
- vm.c (rb_thread_mark): mark cfp->me.
- vm_core.h (rb_thread_t): add a field passed_me.
- vm_core.h (rb_vm_t): add a field unlinked_method_entry_list.
- vm_insnhelper.c (vm_push_frame): clear cfp->me at all times.
- vm_insnhelper.c (vm_call_bmethod): pass me.
- bootstraptest/test_method.rb: add a test.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@27634 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 02/22/2010 04:14 PM - ko1 (Koichi Sasada)

=begin
(2010/02/22 15:03), Nobuyoshi Nakada wrote::

> rb_method_entry_tにもrefcountを入れますかねぇ。

□性能の観点から反対します.

□さっきの IRC に書いた内容ですが，こんな感じでどうでしょうか. 大仰な気もしますが.

(1) unlink された me を保存しておく
(2) 一定のタイミング（多分，GCタイミング）でフレームを走査して
□□保存した me に mark
(3) mark がついてないのを消す（sweep）

□NODE に戻すってのも一案ですが....

--
// SASADA Koichi at atdot dot net

=end

#2 - 04/20/2010 09:33 PM - mame (Yusuke Endoh)

- Assignee set to ko1 (Koichi Sasada)

- Priority changed from 3 to Normal

- Target version set to 1.9.2

- ruby -v set to -

=begin
遠藤です。

```
% cat tst.rb
module Kernel
def load(*args)
end
end
raise
% valgrind ./ruby -ve 'load "tst.rb"'
snip
```

というオリジナルの問題は r27393 で巧妙に回避されましたが、本質的には
解決しておらず、以下で SEGV します。

```
class C
define_method(:foo) do
C.class_eval { remove_method(:foo) }
super()
end
end
C.new.foo
```

あと、r27393 だと rb_method_definition_t の参照を減らしていないので
メモリリークするような気がします。

```
diff --git a/vm_method.c b/vm_method.c
index 04b62f2..c9d99db 100644
--- a/vm_method.c
+++ b/vm_method.c
@@ -215,6 +215,14 @@ rb_add_method_def(VALUE klass, ID mid, rb_method_type_t type, rb_method_definiti
* another problem when the usage is changed.
*/
me = old_me;
+
- if (me->def) {
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- }
  }
  else {
  me = ALLOC(rb_method_entry_t);
```

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

#3 - 05/06/2010 02:57 AM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
This issue was solved with changeset r27634.
Nobuyoshi, thank you for reporting this issue.

Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end