Ruby - Bug #9321

rb_mod_const_missing does not generate a c-return event

12/30/2013 06:23 PM - drkaes (Stefan Kaes)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.1.0p1 (2013-12-26 revision 44431) [x86_64-darwin12.0]	**Backport:**	2.0.0: DONE, 2.1: DONE

Description

We have had an issue reported for ruby-prof where execution times were attributed incorrectly in the call graph.

It turned out that the problem is caused by a missing c-return event for Module#const_missing.

ruby-prof simulates the ruby call stack by subscribing to line, call, c-call, return and c-return events.

Obviously, the missing return throws ruby-prof off the track.

Associated revisions

Revision c2e72fb34302e444f6cc4c6477877e36902a7035 - 01/09/2014 10:12 AM - ko1 (Koichi Sasada)

- vm.c (rb_vm_pop_cfunc_frame): added. It cares c_return event.
  The patch base by drkaes (Stefan Kaes).
  [Bug #9321]
- variable.c (rb_mod_const_missing): use rb_vm_pop_cfunc_frame()
  instead of rb_frame_pop().
- vm_eval.c (raise_method_missing): ditto.
- vm_eval.c (rb_iterate): ditto.
- internal.h (rb_vm_pop_cfunc_frame): add decl.
- test/ruby/test_settracefunc.rb: add tests.
  provided by drkaes (Stefan Kaes).
- vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
  move definition of rb_frame_pop() and deprecate it.
  It doesn't care about `return' events.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@44535 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision c2e72fb3 - 01/09/2014 10:12 AM - ko1 (Koichi Sasada)

- vm.c (rb_vm_pop_cfunc_frame): added. It cares c_return event.
  The patch base by drkaes (Stefan Kaes).
  [Bug #9321]
- variable.c (rb_mod_const_missing): use rb_vm_pop_cfunc_frame()
  instead of rb_frame_pop().
- vm_eval.c (raise_method_missing): ditto.
- vm_eval.c (rb_iterate): ditto.
- internal.h (rb_vm_pop_cfunc_frame): add decl.
- test/ruby/test_settracefunc.rb: add tests.
  provided by drkaes (Stefan Kaes).
- vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
  move definition of rb_frame_pop() and deprecate it.
  It doesn't care about `return' events.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@44535 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision bdf635bf5be3213770c8a77380a045e2ac1c19af - 06/29/2014 04:13 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) r44535,r44536: [Backport #9321]

```
    * vm.c (rb_vm_pop_cfunc_frame): added.  It cares c_return event.
      The patch base by drkaes (Stefan Kaes).
      [Bug #9321]

    * variable.c (rb_mod_const_missing): use rb_vm_pop_cfunc_frame()
```

```
      instead of rb_frame_pop().

    * vm_eval.c (raise_method_missing): ditto.

    * vm_eval.c (rb_iterate): ditto.

    * internal.h (rb_vm_pop_cfunc_frame): add decl.

    * test/ruby/test_settracefunc.rb: add tests.
      provided by drkaes (Stefan Kaes).

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
      move definition of rb_frame_pop() and deprecate it.
      It doesn't care about `return' events.

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_1@46608 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision bdf635bf - 06/29/2014 04:13 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) r44535,r44536: [Backport #9321]

```
    * vm.c (rb_vm_pop_cfunc_frame): added.  It cares c_return event.
      The patch base by drkaes (Stefan Kaes).
      [Bug #9321]

    * variable.c (rb_mod_const_missing): use rb_vm_pop_cfunc_frame()
      instead of rb_frame_pop().

    * vm_eval.c (raise_method_missing): ditto.

    * vm_eval.c (rb_iterate): ditto.

    * internal.h (rb_vm_pop_cfunc_frame): add decl.

    * test/ruby/test_settracefunc.rb: add tests.
      provided by drkaes (Stefan Kaes).

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
      move definition of rb_frame_pop() and deprecate it.
      It doesn't care about `return' events.

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_1@46608 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 6ece90852b3b12a8873d85b35104a93867dbfc14 - 07/03/2014 06:20 AM - U.Nakamura

merge revision(s) 44535,44536: [Backport #9321]

```
    * vm.c (rb_vm_pop_cfunc_frame): added.  It cares c_return event.
      The patch base by drkaes (Stefan Kaes).
      [Bug #9321]

    * variable.c (rb_mod_const_missing): use rb_vm_pop_cfunc_frame()
      instead of rb_frame_pop().

    * vm_eval.c (raise_method_missing): ditto.

    * vm_eval.c (rb_iterate): ditto.

    * internal.h (rb_vm_pop_cfunc_frame): add decl.

    * test/ruby/test_settracefunc.rb: add tests.
      provided by drkaes (Stefan Kaes).

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
      move definition of rb_frame_pop() and deprecate it.
      It doesn't care about `return' events.

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_0_0@46671 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 6ece9085 - 07/03/2014 06:20 AM - U.Nakamura

merge revision(s) 44535,44536: [Backport #9321]

```
    * vm.c (rb_vm_pop_cfunc_frame): added.  It cares c_return event.
      The patch base by drkaes (Stefan Kaes).
      [Bug #9321]

    * variable.c (rb_mod_const_missing): use rb_vm_pop_cfunc_frame()
      instead of rb_frame_pop().

    * vm_eval.c (raise_method_missing): ditto.

    * vm_eval.c (rb_iterate): ditto.

    * internal.h (rb_vm_pop_cfunc_frame): add decl.

    * test/ruby/test_settracefunc.rb: add tests.
      provided by drkaes (Stefan Kaes).

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
      move definition of rb_frame_pop() and deprecate it.
      It doesn't care about `return' events.

    * vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_0_0@46671 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 12/30/2013 06:28 PM - drkaes (Stefan Kaes)

- File rb_mod_const_missing.patch added

I believe the attached patch fixes the problem.

#2 - 12/30/2013 08:05 PM - drkaes (Stefan Kaes)

- File rb_mod_const_missing_with_tests.patch added

added tests to patch.

#3 - 12/30/2013 08:06 PM - drkaes (Stefan Kaes)

I believe this bug also exists in 1.9.3 and 2.0.0.

Any chance to backport?

#4 - 12/30/2013 08:55 PM - tmm1 (Aman Karmani)

Thanks for adding the test.

Seems like rb_frame_pop() should probably emit the event, but I'm not sure how to tell klass/mid in there.

BTW, doesn't this problem happen anytime an exception is raised? One or more stack frames can disappear without corresponding events.

#5 - 12/30/2013 09:21 PM - drkaes (Stefan Kaes)

rb_frame_pop is called in only one place, and that's the one which the patch addresses.

So yes, we could move the code to rb_frame_pop(), if we change it's signature.

I'm not sure, under which circumstances the problem can show it's ugly head.

This breaks (without patch):

def foo; UNKNOWN_CONSTANT rescue nil; end

These don't:

def foo; 1/0 rescue nil; end
def bar; raise "shoot" rescue nil; end

#6 - 12/31/2013 06:55 PM - drkaes (Stefan Kaes)

The patch has some issues with redefined Module#const_missing.

I'm in the process of preparing a new one.

#7 - 12/31/2013 07:35 PM - drkaes (Stefan Kaes)

Here's the new patch. I have moved the hook to rb_frame_pop().

The new patch makes sure that the method names sent along with call and return events are identical.

But I'm not sure whether the current ruby behavior in the presence of aliased methods is really the desired behavior.

Looking forward to feedback on this topic.

#8 - 12/31/2013 07:39 PM - drkaes (Stefan Kaes)

- File rb_mod_const_missing_with_tests_redefined.patch added

#9 - 12/31/2013 09:24 PM - ko1 (Koichi Sasada)

The last patch is not acceptable because of performance reason.

I'll think about it.

#10 - 01/01/2014 08:31 AM - tmm1 (Aman Karmani)

I like the second patch, but now I'm not so sure about this technique.

Technically, const_missing never returns. So emitting a c_return event seems wrong. In fact, a raise event should be emitted from inside const_missing.

The "right" way to deal with this in ruby-prof would be to listen for the raise event and unwind the internal representation of the stack accordingly. Unfortunately, this is kind of tricky to implement.

#11 - 01/02/2014 08:30 PM - tmm1 (Aman Karmani)

In rblineprof, I maintain a stack of CALL events. When a RETURN event comes in, CALL entries on the internal stack are popped until one matching the RETURN is found.

https://github.com/tmm1/rblineprof/blob/32795afc748f2f51cf76ce20ef6a85f50b9278c6/ext/rblineprof.c#L471-L475

#12 - 01/03/2014 03:19 AM - drkaes (Stefan Kaes)

@ko1 (Koichi Sasada): Why do you think the second patch creates a performance problem?

Module#const_missing is an exceptional case. Am I missing something?

@tmm1 (Aman Karmani): the stack tracking approach breaks down if the method id reported on return by ruby does not match the one reported on call; which I have seen happening in the past.

#13 - 01/03/2014 02:38 PM - drkaes (Stefan Kaes)

@tmm1's suspicion was right: c-return is also missing in the case when you try to call a nonexistent method.

It seems that c-return will be missing when raising an exception from C land.

Which means my patch doesn't really solve the underlying, more fundamental problem.

#14 - 01/04/2014 02:10 AM - deivid (David Rodríguez)

But when raising an exception from ruby, return events are emitted... Shouldn't this 2 cases be consistent?

#15 - 01/06/2014 03:56 AM - deivid (David Rodríguez)

In byebug, the problem is I don't manually mantain the call stack, but only keep track of the stack size through the TracePoint API events. The only time I load the call stack is through the DebugInspector API right before the debugger stops in order to make it available to the user.

If I was to implement the solution suggested by @tmm1 (Aman Karmani), how can I access the method_id for each frame from the C extension? Is the Thread::Backtrace::Locations API available to C extensions? Calling it through ruby for every c_return event will impact performance I guess...

#16 - 01/06/2014 05:51 AM - tmm1 (Aman Karmani)

You can use rb_profile_frames() in 2.1, but it only reports ruby frames so it won't help identify C-call/return events.

You could also use TracePoint#method_id either from ruby or C, but @skaes reports the method ids sometimes don't match up. Maybe we can figure out why and fix it.

#17 - 01/06/2014 04:25 PM - drkaes (Stefan Kaes)

I have extended the patch to also cover method_missing, by changing a single cfp pointer manipulation to a call of rb_frame_pop().

See https://github.com/skaes/rvm-patchsets/blob/master/patches/ruby/2.1.0/head/railsexpress/05-fix-missing-c-return-event.patch#L121

I have tested various other c functions raising exceptions. All of them work fine without the patches.

And my position is still that this is a ruby bug, which should be fixed instead of being worked around by all tool implementers.

The only thing stopping us from fixing it is @ko1's comment on performance impact, which I don't understand.

#18 - 01/06/2014 04:38 PM - drkaes (Stefan Kaes)

The patch isn't ideal yet: in case of const-missing, the c-return event is generated too early. I know how to fix this. Will upload a modified patch later.

#19 - 01/07/2014 05:04 PM - deivid (David Rodríguez)

@drkaes If this only happens in certain c-methods it certainly looks like a bug. Thanks for working on this.

Thanks @tmm1 (Aman Karmani), I guess I would have to store the method_id provided by the TracePoint API to be able to match it later.

#20 - 01/07/2014 06:23 PM - ko1 (Koichi Sasada)

(2014/01/03 3:19), drkaes (Stefan Kaes) wrote:

> @ko1 (Koichi Sasada): Why do you think the second patch creates a performance problem?
>
> Module#const_missing is an exceptional case. Am I missing something?

Sorry for my misunderstanding.

Let me clarify the spec.

```
###
n = 0
TracePoint.new(){|tp|
  next if tp.event == :line
  case tp.event.to_s
  when /call/
    puts "#{' ' * 2 * n}#{tp.inspect}"
    n+=1
  when /return/
    n-=1 if n>0
    puts "#{' ' * 2 * n}#{tp.inspect}"
  else
    puts "#{' ' * 2 * n}#{tp.inspect}"
  end
}.enable

def f
  Object::XYZZY #=> const_missing
end

f
```
__END__
```
#=>

current behavior:

#<TracePoint:c_return `enable'@test.rb:15>
#<TracePoint:c_call `method_added'@test.rb:17>
#<TracePoint:c_return `method_added'@test.rb:17>
#<TracePoint:call `f'@test.rb:17>
  #<TracePoint:c_call `const_missing'@test.rb:18>
    #<TracePoint:c_call `initialize'@test.rb:18>
      #<TracePoint:c_call `initialize'@test.rb:18>
      #<TracePoint:c_return `initialize'@test.rb:18>
    #<TracePoint:c_return `initialize'@test.rb:18>
    #<TracePoint:c_call `exception'@test.rb:18>
    #<TracePoint:c_return `exception'@test.rb:18>
    #<TracePoint:c_call `backtrace'@test.rb:18>
    #<TracePoint:c_return `backtrace'@test.rb:18>
    #<TracePoint:raise@test.rb:18>
  #<TracePoint:return `f'@test.rb:18>

expected behavior:

#<TracePoint:c_return `enable'@test.rb:15>
#<TracePoint:c_call `method_added'@test.rb:17>
#<TracePoint:c_return `method_added'@test.rb:17>
#<TracePoint:call `f'@test.rb:17>
  #<TracePoint:c_call `const_missing'@test.rb:18>
  #<TracePoint:c_return `const_missing'@test.rb:18> <-- NEW!
  #<TracePoint:c_call `initialize'@test.rb:18>
    #<TracePoint:c_call `initialize'@test.rb:18>
    #<TracePoint:c_return `initialize'@test.rb:18>
  #<TracePoint:c_return `initialize'@test.rb:18>
  #<TracePoint:c_call `exception'@test.rb:18>
  #<TracePoint:c_return `exception'@test.rb:18>
  #<TracePoint:c_call `backtrace'@test.rb:18>
  #<TracePoint:c_return `backtrace'@test.rb:18>
  #<TracePoint:raise@test.rb:18>
#<TracePoint:return `f'@test.rb:18>
```

is it right?

This behavior is similar to "raise" behavior.

--
// SASADA Koichi at atdot dot net

#21 - 01/07/2014 06:53 PM - ko1 (Koichi Sasada)

I modify the patch and all tests passes.
http://www.atdot.net/sp/raw/6ny0zm

If we can accept the spec of this behavior (order of tracing events), I
will commit it.

--
// SASADA Koichi at atdot dot net

#22 - 01/07/2014 08:06 PM - drkaes (Stefan Kaes)

The order of trace events is acceptable.

It's not ideal though: I think one would expect the events for creating the exception and attaching the backtrace to it to happen inside const_missing/method_missing.

But it seems to me this would require more complex code changes.

Thx a lot for fixing this.

Is it possible to backport this to 2.0 and 1.9.3?

#23 - 01/07/2014 08:10 PM - deivid (David Rodríguez)

It sounds good to me. Any behavior with a balanced count of call and return events work.

Big thanks @ko1 (Koichi Sasada).

#24 - 01/07/2014 10:14 PM - drkaes (Stefan Kaes)

Compiling with clang produces warnings:

in file included from ./include/ruby/ruby.h:1694:
./include/ruby/intern.h:955:40: warning: unknown attribute 'warning' ignored [-Wattributes]
void rb_frame_pop(void) **attribute**((warning("rb_frame_pop is obsolete. It will be deleted after Ruby 2.2.0.")));;

You probably need to change the method signature to:

DEPRECATED(void rb_frame_pop(void));

#25 - 01/09/2014 07:13 PM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r44535.
Stefan, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- vm.c (rb_vm_pop_cfunc_frame): added. It cares c_return event.
  The patch base by drkaes (Stefan Kaes).
  [Bug #9321]
- variable.c (rb_mod_const_missing): use rb_vm_pop_cfunc_frame()
  instead of rb_frame_pop().
- vm_eval.c (raise_method_missing): ditto.
- vm_eval.c (rb_iterate): ditto.
- internal.h (rb_vm_pop_cfunc_frame): add decl.
- test/ruby/test_settracefunc.rb: add tests.
  provided by drkaes (Stefan Kaes).
- vm.c, eval.c, include/ruby/intern.h (rb_frame_pop):
  move definition of rb_frame_pop() and deprecate it.
  It doesn't care about `return' events.

#26 - 03/30/2014 04:32 PM - deivid (David Rodríguez)

Hi @koichi, I'm still running into this issue... :(I really don't know what's going on, because the tests pass in ruby's test suite, but the c_return event is not generated when I use the TracePoint API.

```
require 'minitest/autorun'

class TestTracePoint < Minitest::Test
  def test_method_missing
    events = []
    assert !respond_to?(:missing_method_59398)
    TracePoint.new(:c_call, :c_return, :call, :return) { |tp|
      events << [tp.event, tp.method_id] if tp.method_id == :method_missing
    }.enable {
      missing_method_59398 rescue nil
    }
    assert_equal([[:c_call, :method_missing], [:c_return, :method_missing]], events)
  end
end

$ ruby --version
ruby 2.1.1p76 (2014-02-24 revision 45161) [i686-linux]

$ ruby test_tracepoint.rb
Run options: --seed 31045

# Running:

F

Finished in 0.035420s, 28.2322 runs/s, 56.4645 assertions/s.

  1) Failure:
TestTracePoint#test_method_missing [test_tracepoint.rb:12]:
--- expected
+++ actual
@@ -1 +1 @@
-[[:c_call, :method_missing], [:c_return, :method_missing]]
+[[:c_call, :method_missing]]


1 runs, 2 assertions, 1 failures, 0 errors, 0 skips
```

Also I have a couple of failing tests in byebug's test suite because of this. https://travis-ci.org/deivid-rodriguez/byebug/jobs/21874423

Thanks a lot.

#27 - 03/30/2014 10:38 PM - ko1 (Koichi Sasada)

- Status changed from Closed to Open

#28 - 03/31/2014 09:31 AM - ko1 (Koichi Sasada)

I check with test/unit (modified as follow) and I got no error. Do I miss anything?

```
require 'test/unit'

class TestTracePoint < Test::Unit::TestCase
  def test_method_missing
    events = []
    assert !respond_to?(:missing_method_59398)
    TracePoint.new(:c_call, :c_return, :call, :return) { |tp|
      events << [tp.event, tp.method_id] if tp.method_id == :method_missing
    }.enable {
      missing_method_59398 rescue nil
    }

    assert_equal([[:c_call, :method_missing], [:c_return, :method_missing]], events)
  end
end
```

And I got

```
1 tests, 2 assertions, 0 failures, 0 errors, 0 skips
```

```
ruby -v: ruby 2.2.0dev (2014-03-31 trunk 45486) [x86_64-linux]
```

Thanks,
Koichi

#29 - 03/31/2014 02:10 PM - deivid (David Rodríguez)

no no I was missing something.... :(I thought this bug fix was included in 2.1 but it's not. Last master works fine. I'm sorry for the confusion. And thanks a lot for having a look so quick!

Can I ask for this to be backported to 2.0 and 2.1?

Thanks!

#30 - 04/01/2014 05:19 AM - ko1 (Koichi Sasada)

Ah, I see.

I think it should be backported, but it changes a behavior.

@naruse-san, can we backport it?

#31 - 04/01/2014 06:07 AM - naruse (Yui NARUSE)

- Status changed from Open to Closed

- Backport changed from 1.9.3: UNKNOWN, 2.0.0: UNKNOWN, 2.1: UNKNOWN to 2.0.0: REQUIRED, 2.1: REQUIRED

□□□□□□□□□

> const_missing □□□□□□□□c_call □□□□□□□□ c_return □□□□□□□□□□□□□□□□□□□□□□□□□□□□
> c_return □□□□□□□□□□□□
> □□□□□□□□□□□□□□□□□□□□□□□□□□□□
> □□□□□□
> □□□□□□□□□□□□□□□□
> https://bugs.ruby-lang.org/issues/9321#note-20

#32 - 04/01/2014 06:13 AM - ko1 (Koichi Sasada)

This issues is available on 2.1 and 2.0.

This is possible that some applications implement workaround for this issue. This patch can breaks such applications. However, I'm not sure such applications exist (maybe don't exist).

#33 - 04/01/2014 11:45 AM - deivid (David Rodríguez)

As far as I'm concerned, this changes an (incorrect) behaviour and it should be backported. In byebug's case, having a balanced count of call and return events is critical and directly affects usability of the debugger. And when I thought of implementing a workaround inside byebug's didn't seem like an easy thing to do...

Stefan Kaes, did you implement a workaround for this in ruby-prof?

Cheers!

#34 - 04/14/2014 10:10 PM - ko1 (Koichi Sasada)

I vote to backport this ticket. I want to help them:
https://github.com/deivid-rodriguez/byebug/issues/16#issuecomment-40421841

#35 - 04/17/2014 07:54 AM - deivid (David Rodríguez)

Thanks Koichi, appreciated. :)

#36 - 06/09/2014 01:05 PM - deivid (David Rodríguez)

Is there an official way to request a backport? Thanks!

#37 - 06/19/2014 06:08 AM - nagachika (Tomoyuki Chikanaga)

I agree that this should be backported to 2.1 branch.
Is there any opposite opinion?

#38 - 06/23/2014 10:43 PM - deivid (David Rodríguez)

Thanks for following up Tomoyuki!

#39 - 06/29/2014 04:13 PM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.0.0: REQUIRED, 2.1: REQUIRED to 2.0.0: REQUIRED, 2.1: DONE

Backported into ruby_2_1 branch at r46608.

#40 - 07/03/2014 06:21 AM - usa (Usaku NAKAMURA)

backported into ruby_2_0_0 at r46671.

#41 - 07/03/2014 06:25 AM - usa (Usaku NAKAMURA)

- Backport changed from 2.0.0: REQUIRED, 2.1: DONE to 2.0.0: DONE, 2.1: DONE

Files

rb_mod_const_missing.patch	643 Bytes	12/30/2013	drkaes (Stefan Kaes)
rb_mod_const_missing_with_tests.patch	1.48 KB	12/30/2013	drkaes (Stefan Kaes)
rb_mod_const_missing_with_tests_redefined.patch	3.01 KB	12/31/2013	drkaes (Stefan Kaes)