Ruby master - Bug #7556

test error on refinement

12/13/2012 08:13 PM - usa (Usaku NAKAMURA)

Status: Closed
Priority: Normal
Assignee: ko1 (Koichi Sasada)
Target version: 2.0.0

ruby -v:
ruby -v: ruby 2.0.0dev (2012-12-13 trunk 38354) [x64-mswin64_100]

Description
1) Error:
test_refine_recursion(TestRefinement):
NoMethodError: undefined method recursive_length' for "oo":String
C:/Users/usa/ruby/test/ruby/test_refinement.rb:567:in recursive_length'
    :in <main>
C:/Users/usa/ruby/test/ruby/test_refinement.rb:806:ineval'
C:/Users/usa/ruby/test/ruby/test_refinement.rb:806:in eval_using'
C:/Users/usa/ruby/test/ruby/test_refinement.rb:574:intest_refine_recursion'

On my box this error is 100% reproducible, but I also know that RubyCI and RubyInstaller CI don't report this error.
I've heard that nobu reproduced this bug on x86_64-dawrin, but I don't know the detail of his environment.

Once I wrote the detail of my debuggin, but it is lost by accidenal reboot of my PC.

I have no energy to rewrite it, because writing long English sentences irritates me, especially after seeing mails which reproach our native language.

Associated revisions

Revision 1d7f7375 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

• vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.

• vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@38377 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 38377 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

• vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.

• vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same
method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

Revision 38377 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

- vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.
- vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

Revision 38377 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

- vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.
- vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

Revision 38377 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

- vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.
- vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

Revision 38377 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

- vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.
- vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

Revision 38377 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

- vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.
- vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

Revision 38377 - 12/14/2012 08:04 AM - shugo (Shugo Maeda)

- vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.
- vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

History

08/07/2021
On 2012/12/13 20:13, usa (Usaku NAKAMURA) wrote:

Issue #7556 has been reported by usa (Usaku NAKAMURA).

Once I wrote the detail of my debuggin, but it is lost by accidental reboot of my PC.

I have no energy to rewrite it, because writing long English sentences irritates me, especially after seeing mails which reproach our native language.

Two solutions:

1) Write short English sentences.
2) Write in Japanese. (If somebody thinks they really need to know what you wrote, they can ask on the list.)

But I know that for creative people, it is much harder to do the same work again than to do it the first time. I once worked on Devanagari rendering, but lost that work in a reboot. I didn't want to redo it, so I worked on Tamil instead. (I just wanted to see whether my architecture was able to handle Indic rendering issues.)

Regards, Martin.

#2 - 12/13/2012 10:32 PM - shugo (Shugo Maeda)
- Assignee changed from shugo (Shugo Maeda) to ko1 (Koichi Sasada)

usa (Usaku NAKAMURA) wrote:

1) Error:
   test_refine_recursion(TestRefinement):
   NoMethodError: undefined method 'recursive_length' for "oo":String
   C:/Users/usa/ruby/test/ruby/test_refinement.rb:567:in 'recursive_length'
   :in <main>
   C:/Users/usa/ruby/test/ruby/test_refinement.rb:806:ineval'
   C:/Users/usa/ruby/test/ruby/test_refinement.rb:806:in eval_using'
   C:/Users/usa/ruby/test/ruby/test_refinement.rb:574:intest_refine_recursion'

To avoid an infinite loop by super in a refinement, ci->call is used to distinguish super calls from normal calls, and if it's a super call, skip the same method.
However, VC++ optimizes vm_call_general and vm_call_super_method into the same method because they have the same definition, so ci->call cannot be used to distinguish super calls from normal calls. How intelligent VC++ is!

I've found that the following hack fixes the problem:

    static VALUE
    vm_call_super_method(rb_thread_t th, rb_control_frame_t *reg_cfp, rb_call_info_t *ci)
    {
      #ifdef _WIN32
      volatile int x = 0; /* to avoid VC++ optimization which makes
      vm_call_super_method as an alias of
      vm_call_general! */
      #endif
      return vm_call_method(th, reg_cfp, ci);
    }

Sasada-san, do you accept this ugly hack, or do you come up with a neat solution?

#3 - 12/13/2012 11:23 PM - ko1 (Koichi Sasada)

(2012/12/13 22:32), shugo (Shugo Maeda) wrote:
Sasada-san, do you accept this ugly hack, or do you come up with a neat solution?

To answer your question, I need to learn how refinement is implemented. Please wait a moment. (or please commit it ahead and left this ticket open)

... // SASADA Koichi at atdot dot net

#4 - 12/13/2012 11:23 PM - shugo (Shugo Maeda)

Hi,

2012/12/13 SASADA Koichi ko1@atdot.net:

Sasada-san, do you accept this ugly hack, or do you come up with a neat solution?

To answer your question, I need to learn how refinement is implemented. Please wait a moment. (or please commit it ahead and left this ticket open)

Thanks for your quick response. I've committed the workaround, and have left the ticket open.

... Shugo Maeda

#5 - 12/14/2012 12:44 AM - phasis68 (Heesob Park)

Here is another workaround:

```c
#ifdef _MSC_VER
#pragma optimize( "", off )
#endif
static VALUE
vm_call_super_method(rb_thread_t *th, rb_control_frame_t *reg_cfp, rb_call_info_t *ci)
{
    return vm_call_method(th, reg_cfp, ci);
}
#endif
#pragma optimize( "", on )
#endif
```

#6 - 12/14/2012 01:47 PM - shugo (Shugo Maeda)

Hello,

phasis68 (Heesob Park) wrote:

Here is another workaround:

```c
#ifdef _MSC_VER
#pragma optimize( "", off )
#endif
```

Thanks for your suggestion.

But it seems that the `#pragma optimize( "", off )` version is slightly slower than the volatile int `x = 0` version.

with volatile int `x = 0`:

```
C:\Users\shugo\Documents\Source\ruby>ruby\bin\ruby bm_vm2_super.rb
Rehearsal
super 1.778000 0.000000 1.778000 ( 1.783227)
------------------------- total: 1.778000sec
            user    system   total       real
super 1.810000 0.000000  1.810000 ( 1.806230)
```

C:\Users\shugo\Documents\Source\ruby>ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super   1.747000   0.000000   1.747000 (  1.789727)  
-------------------------------- total: 1.747000sec

user system total real
super   1.763000   0.000000   1.763000 (  1.760224)

C:\Users\shugo\Documents\Source\ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super   1.763000   0.000000   1.763000 (  1.804229)  
-------------------------------- total: 1.763000sec

user system total real
super   1.809000   0.000000   1.809000 (  1.841734)

with #pragma optimize( "", off ):

C:\Users\shugo\Documents\Source\ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super   1.825000   0.000000   1.825000 (  1.859236)  
-------------------------------- total: 1.825000sec

user system total real
super   1.872000   0.000000   1.872000 (  1.864237)

C:\Users\shugo\Documents\Source\ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super   1.919000   0.000000   1.919000 (  1.920744)  
-------------------------------- total: 1.919000sec

user system total real
super   1.794000   0.000000   1.794000 (  1.820232)

C:\Users\shugo\Documents\Source\ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super   1.872000   0.000000   1.872000 (  1.913243)  
-------------------------------- total: 1.872000sec

user system total real
super   1.810000   0.000000   1.810000 (  1.817231)

I guess #pragma optimize( "", off ) disables function inlining of vm_call_method.

#7 - 12/14/2012 02:51 PM - phasis68 (Heesob Park)
Here is a modified version which does not disable function inline expansion.

#define _MSC_VER
#pragma optimize("g",off)
#endif

static VALUE
vm_call_super_method(rb_thread_t *th, rb_control_frame_t *reg_cfp, rb_call_info_t *ci)
{
  return vm_call_method(th, reg_cfp, ci);
}
#define _MSC_VER
#pragma optimize("g",on)
#endif

#8 - 12/14/2012 03:13 PM - shugo (Shugo Maeda)
phasis68 (Heesob Park) wrote:

Here is a modified version which does not disable function inline expansion.

#define _MSC_VER
#pragma optimize("g",off)
#endif

08/07/2021
I also tried it, but couldn't see improvement:

```
C:\Users\shugo\Documents\Source\ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super  1.856000   0.000000   1.856000 (  1.868237)
-------------------------------- total: 1.856000sec

user system total     real
super  1.872000   0.000000   1.872000 (  1.910743)
```

```
C:\Users\shugo\Documents\Source\ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super  1.841000   0.000000   1.841000 (  1.860736)
-------------------------------- total: 1.841000sec

user system total     real
super  1.903000   0.016000   1.919000 (  1.956249)
```

```
C:\Users\shugo\Documents\Source\ruby\bin\ruby bm_vm2_super.rb
Rehearsal -----------------------------------------
super  1.872000   0.000000   1.872000 (  1.927244)
-------------------------------- total: 1.872000sec

user system total     real
super  1.841000   0.000000   1.841000 (  1.868237)
```

And, other options of the optimize pragma such as "p" don't fix the problem.

And, other options of the optimize pragma such as "p" don't fix the problem.

```
#9 - 12/14/2012 03:59 PM - phasis68 (Heesob Park)
Here is a different workaround using __forceinline on vm_call_method function.
```

```
static
#ifdef _MSC_VER
__forceinline
#else
inline
#endif
VALUE
vm_call_method(rb_thread_t *th, rb_control_frame_t *cfp, rb_call_info_t *ci)
{
...
}
```

```
#10 - 12/14/2012 04:55 PM - shugo (Shugo Maeda)
phasis68 (Heesob Park) wrote:
Here is a different workaround using __forceinline on vm_call_method function.
```

```
static
#ifdef _MSC_VER
__forceinline
#else
inline
#endif
VALUE
vm_call_method(rb_thread_t *th, rb_control_frame_t *cfp, rb_call_info_t *ci)
```

Thanks, it solves the problem without the optimize pragma for rb_call_super_method, and performance decrease hasn't been observed.

Could you tell me why __forceinline for vm_call_method prevent VC++ to make vm_call_general and vm_call_super_method as the same function?
I couldn't find the reason at URL: http://msdn.microsoft.com/library/vstudio/z8y1yy88.

```
#11 - 12/14/2012 05:04 PM - shugo (Shugo Maeda)
- Status changed from Assigned to Closed
- % Done changed from 0 to 100
```

This issue was solved with changeset r38377.
Usaku, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- vm_insnhelper.c (vm_call_super_method): remove volatile introduced in r38365.
- vm_insnhelper.c (vm_call_method): use __forceinline to prevent VC to make vm_call_general and vm_call_super_method as the same method. Thanks, Heesob Park. [Bug #7556] [ruby-core:50867]

#12 - 12/14/2012 05:41 PM - phasis68 (Heesob Park)
It seems that inline or __inline is not respected by the compiler (ignored by compiler cost/benefit analyzer)
Refer to http://en.wikipedia.org/wiki/Inline_function

#13 - 12/14/2012 05:57 PM - shugo (Shugo Maeda)
phasis68 (Heesob Park) wrote:

- It seems that inline or __inline is not respected by the compiler (ignored by compiler cost/benefit analyzer)
Refer to http://en.wikipedia.org/wiki/Inline_function

I know it, but I don't know why __forceinline prevent user functions (in this case, vm_call_general and vm_call_super_method) to be optimized into a single function.

#14 - 12/14/2012 06:29 PM - phasis68 (Heesob Park)
2012/12/14 shugo (Shugo Maeda) redmine@ruby-lang.org

Issue #7556 has been updated by shugo (Shugo Maeda).

phasis68 (Heesob Park) wrote:

- It seems that inline or __inline is not respected by the compiler (ignored by compiler cost/benefit analyzer)
Refer to http://en.wikipedia.org/wiki/Inline_function

I know it, but I don't know why __forceinline prevent user functions (in this case, vm_call_general and vm_call_super_method) to be optimized into a single function.

The __forceinline keyword overrides the cost/benefit analysis and relies on the judgment of the programmer instead.
Using __forceinline insures that all functions which call vm_call_method function have the inline expanded code instead of vm_call_method function call.
Thus, vm_call_general and vm_call_super_method are separated function.