Ruby master - Bug #16366

.count on endless range causes infinite loop

11/25/2019 03:28 PM - duffyjp (Jacob Duffy)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.5p114 (2019-10-01 revision 67812) [x86_64-darwin19]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN

Description

Out of curiosity, I tried:

```
(1..).count
```

Which resulted in an unkillable 100% CPU irb session.

I expected either an exception or maybe Infinity

Thanks

Associated revisions

Revision 36da0b3d - 11/29/2019 08:47 AM - ko1 (Koichi Sasada)

check interrupts at each frame pop timing.

Asynchronous events such as signal trap, finalization timing,
thread switching and so on are managed by "interrupt_flag".
Ruby's threads check this flag periodically and if a thread
does not check this flag, above events doesn't happen.

This checking is CHECK_INTS() (related) macro and it is placed
at some places (laeve instruction and so on). However, at the end
of C methods, C blocks (IMEMO_IFUNC) etc there are no checking
and it can introduce uninterruptible thread.

To modify this situation, we decide to place CHECK_INTS() at
vm_pop_frame(). It increases interrupt checking points.
[Bug #16366]

This patch can introduce unexpected events...

Revision 00bbdf44 - 12/04/2019 06:32 AM - shyouhei (Shyouhei Urabe)

implement Range#count

As matz requested in [Bug #16366].

History

#1 - 11/25/2019 04:37 PM - shevegen (Robert A. Heiler)

Sounds like a bug - counting to infinity is not easy. :)

#2 - 11/26/2019 01:09 AM - shyouhei (Shyouhei Urabe)

I don't think infinite loop is a bug, but having no chance to ^C is definitely wrong. Must be able to interrupt.

```
zsh % gdb --args ./miniruby -e '(1..).count'
GNU gdb (Ubuntu 8.2-0ubuntu1~18.04) 8.2
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
```

```
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./miniruby...done.
(gdb) run
Starting program: miniruby -e \(1..\).count
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
^C
Program received signal SIGINT, Interrupt.
0x00005555557ed8f3 in rb_yield (val=val@entry=205343759) at vm_core.h:1455
1455        return captured;
(gdb) bt
#0  0x00005555557ed8f3 in rb_yield (val=val@entry=205343759) at vm_core.h:1455
#1  0x00005555557220ee in range_each (range=<optimized out>) at range.c:856
#2  0x00005555557eec10 in vm_call0_cfunc_with_frame (argv=0x0, cd=<optimized out>, calling=<optimized out>, ec
=0x555555b2d650) at vm_eval.c:91
#3  vm_call0_cfunc (argv=0x0, cd=<optimized out>, calling=<optimized out>, ec=0x555555b2d650) at vm_eval.c:105
#4  vm_call0_body (ec=0x555555b2d650, calling=<optimized out>, cd=<optimized out>, argv=0x0) at vm_eval.c:140
#5  0x00005555557f179e in rb_vm_call0 (kw_splat=<optimized out>, me=<optimized out>, argv=<optimized out>, arg
c=<optimized out>, id=2993, recv=93824998431560, ec=0x555555b2d650) at vm_eval.c:52
#6  rb_vm_call_kw (kw_splat=<optimized out>, me=0x555555b6ef90, argv=<optimized out>, argc=<optimized out>, id
=2993, recv=93824998431560, ec=0x555555b2d650) at vm_eval.c:271
#7  rb_call0 (ec=0x555555b2d650, recv=93824998431560, mid=2993, argc=0, argv=0x0, call_scope=<optimized out>, 
self=93824998431560) at vm_eval.c:395
#8  0x00005555557f27b5 in rb_call (scope=<optimized out>, argv=<optimized out>, argc=<optimized out>, mid=<opt
imized out>, recv=<optimized out>) at vm_eval.c:721
#9  iterate_method (obj=obj@entry=140737488339744) at vm_eval.c:1469
#10 0x00005555557e6bd3 in rb_iterate0 (it_proc=it_proc@entry=0x5555557f2780 <iterate_method>, data1=data1@entr
y=140737488339744, ifunc=0x555555b3dad0, ec=ec@entry=0x555555b2d650) at vm_eval.c:1418
#11 0x00005555557e6f07 in rb_iterate (data2=140737488339744, bl_proc=0x55555560ab50 <count_all_i>, data1=14073
7488339744, it_proc=0x5555557f2780 <iterate_method>) at internal.h:1197
#12 rb_block_call (obj=obj@entry=93824998431560, mid=mid@entry=2993, argc=argc@entry=0, argv=argv@entry=0x0, b
l_proc=bl_proc@entry=0x55555560ab50 <count_all_i>, data2=data2@entry=93824998431480)
    at vm_eval.c:1483
#13 0x000055555560916f in enum_count (argc=<optimized out>, argv=<optimized out>, obj=93824998431560) at enum.
c:248
#14 0x00005555557e58f4 in vm_call_cfunc_with_frame (empty_kw_splat=<optimized out>, cd=0x555555c5a8a0, calling
=<optimized out>, reg_cfp=0x7ffff7fccfa0, ec=0x555555b2d650) at vm_insnhelper.c:2467
#15 vm_call_cfunc (ec=0x555555b2d650, reg_cfp=0x7ffff7fccfa0, calling=<optimized out>, cd=0x555555c5a8a0) at v
m_insnhelper.c:2492
#16 0x00005555557eafeb in vm_call_method (ec=0x555555b2d650, cfp=0x7ffff7fccfa0, calling=<optimized out>, cd=<
optimized out>) at vm_insnhelper.c:3006
#17 0x00005555557f6f36 in vm_sendish (block_handler=0, method_explorer=<optimized out>, cd=0x555555c5a8a0, reg
_cfp=0x7ffff7fccfa0, ec=0x555555b2d650) at vm_insnhelper.c:3975
#18 vm_exec_core (ec=0x555555b2d650, initial=<optimized out>) at insns.def:801
#19 0x00005555557e8f3e in rb_vm_exec (ec=0x555555b2d650, mjit_enable_p=1) at vm.c:1907
#20 0x0000555555620fb9 in rb_ec_exec_node (ec=ec@entry=0x555555b2d650, n=n@entry=0x555555b3dc88) at eval.c:277
#21 0x00005555556255f7 in ruby_run_node (n=0x555555b3dc88) at eval.c:335
#22 0x000055555557f35f in main (argc=<optimized out>, argv=<optimized out>) at main.c:50
(gdb)
```

#3 - 11/26/2019 03:59 AM - jeremyevans0 (Jeremy Evans)

- File enum_count_pass-16366.patch added

Attached is a patch that allows interrupting the iteration. It uses rb_thread_check_ints in each loop iteration for Range#count calls (unless a block is passed, as the block case is already interruptable).

#4 - 11/28/2019 06:28 AM - matz (Yukihiro Matsumoto)

Enumerable#count should aware of interrupts. Besides this case, #count should return Infinity (we may need to define Range#count).

Matz.

#5 - 11/29/2019 08:48 AM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

Applied in changeset git|36da0b3da1aed77e0dffb3f54038f01ff574972b.

check interrupts at each frame pop timing.

Asynchronous events such as signal trap, finalization timing,
thread switching and so on are managed by "interrupt_flag".
Ruby's threads check this flag periodically and if a thread
does not check this flag, above events doesn't happen.

This checking is CHECK_INTS() (related) macro and it is placed
at some places (laeve instruction and so on). However, at the end
of C methods, C blocks (IMEMO_IFUNC) etc there are no checking
and it can introduce uninterruptible thread.

To modify this situation, we decide to place CHECK_INTS() at
vm_pop_frame(). It increases interrupt checking points.
[Bug #16366]

This patch can introduce unexpected events...

#6 - 11/29/2019 08:51 AM - ko1 (Koichi Sasada)

I introduce general solution.

Notes:

- there are no "(1..).count" test. can anyone write it?
- My fix can introduce additional problems because it increases interrupt points. Please report us if you find a problem on your app.
- My fix can introduce additional overhead (checking interrupts). I tried to reduce this overhead by rewriting checking code, however, the original code (no modification code) is fastest :p

Files

enum_count_pass-16366.patch	765 Bytes	11/26/2019	jeremyevans0 (Jeremy Evans)