Ruby trunk - Bug #14834

rb_profile_frames SEGV when PC adjusted on IFUNC

06/08/2018 05:07 AM - kivikakk (Ashe Connor)

Status:	Closed		
Priority:	Normal		
Assignee:	shyouhei (Shyouhei Urabe)		
Target version:			
ruby -v:	ruby 2.6.0dev (2018-06-08 trunk 63606) [x86_64-linux]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

Since r62052, we increment ec->cfp->pc by one pointer width (e.g. 8 bytes) in gc_event_hook_body around the EXEC_EVENT_HOOK call.

This becomes a problem when the hook is on an IFUNC: in this case, pc == 0x0, meaning we increment it to a non-zero value during that call.

rb_profile_frames uses the following check to determine if frame info should be recorded:

```
    if (cfp->iseq && cfp->pc) {
```

The example here is stackprof, which calls rb_profile_frames in a gc event hook. This will segfault currently, as the above check will pass.

calc_lineno then attempts to calculate the line number:

```
        size_t pos = (size_t)(pc - iseq->body->iseq_encoded);
```

This fails for a variety of reasons: iseq_encoded isn't valid because iseq isn't an rb_iseq_t underneath, producing an essentially random value, and pc is 0x8, so we underflow and eventually cause an overrun in succ_index_lookup with a huge pos argument.

We instead only adjust PC if it appears to be a valid pointer in the first place.

Related issues:

Related to Ruby trunk - Bug #14809: calc_lineno() returns wrong location	**Closed**

Associated revisions

Revision 6b534134 - 06/27/2018 09:28 AM - shyouhei (Shyouhei Urabe)

give up insn attr handles_frame

I introduced this mechanism in r62051 to speed things up. Later it
was reported that the change causes problems. I searched for
workarounds but nothing seemed appropriate. I hereby officially
give it up. The idea to move ADD_PC around was a mistake.

Fixes [Bug #14809] and [Bug #14834].

Signed-off-by: Urabe, Shyouhei shyouhei@ruby-lang.org

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63763 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63763 - 06/27/2018 09:28 AM - shyouhei (Shyouhei Urabe)

give up insn attr handles_frame

I introduced this mechanism in r62051 to speed things up. Later it
was reported that the change causes problems. I searched for
workarounds but nothing seemed appropriate. I hereby officially
give it up. The idea to move ADD_PC around was a mistake.

Fixes [Bug #14809] and [Bug #14834].

Signed-off-by: Urabe, Shyouhei shyouhei@ruby-lang.org

Revision 63763 - 06/27/2018 09:28 AM - shyouhei (Shyouhei Urabe)

give up insn attr handles_frame

I introduced this mechanism in r62051 to speed things up. Later it
was reported that the change causes problems. I searched for
workarounds but nothing seemed appropriate. I hereby officially
give it up. The idea to move ADD_PC around was a mistake.

Fixes [Bug #14809] and [Bug #14834].

Signed-off-by: Urabe, Shyouhei shyouhei@ruby-lang.org

History

#1 - 06/08/2018 05:13 AM - kivikakk (Ashe Connor)

It's also worth noting:

- is the increment of pc with ec->cfp->pc++ correct? What if the instruction is multiple values wide?
- there are similar pc increment/decrement pairs around EXEC_EVENT_HOOK calls in vm_trace. Do we need to address these too?

Ping tenderlovemaking (Aaron Patterson) as he helped me with this.

#2 - 06/08/2018 05:44 AM - kivikakk (Ashe Connor)

Further:

- is the original change correct? Some instructions have the handle_frame? property, which means they *do* increment the PC by the instruction width first. It's those that don't have that property which now increment the PC after. The gc hook caller always increments PC, assuming non-handle_frame? instructions.

#3 - 06/08/2018 08:13 AM - shyouhei (Shyouhei Urabe)

- Assignee set to shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

Thanks reporting! Will handle it.

#4 - 06/25/2018 05:47 AM - kivikakk (Ashe Connor)

Here's an easy way to reproduce the segfault on current trunk:

Gemfile

```
source "https://rubygems.org"
gem "stackprof"
```

test.rb

```
require 'stackprof'

StackProf.run(mode: :object) do
  [123].group_by {}
end
```

Output:

```
$ ruby test.rb
test.rb:4: [BUG] Segmentation fault at 0x0000000000000000
ruby 2.6.0dev (2018-06-25 trunk 63747) [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0007 p:---- s:0024 e:000023 IFUNC
c:0006 p:---- s:0021 e:000020 CFUNC  :each
c:0005 p:---- s:0018 e:000017 CFUNC  :group_by
c:0004 p:0007 s:0014 e:000013 BLOCK  test.rb:4 [FINISH]
c:0003 p:---- s:0011 e:000010 CFUNC  :run
c:0002 p:0020 s:0006 e:000005 EVAL   test.rb:3 [FINISH]
c:0001 p:0000 s:0003 E:002330 (none) [FINISH]

-- Ruby level backtrace information ----------------------------------------
test.rb:3:in `<main>'
test.rb:3:in `run'
test.rb:4:in `block in <main>'
```

```
test.rb:4:in `group_by'
test.rb:4:in `each'
Segmentation fault
```

When using gdb:

```
$ gdb -q --args `rbenv which ruby` test.rb

Reading symbols from /home/kivikakk/.rbenv/versions/ruby-trunk/bin/ruby...done.
(gdb) run
Starting program: /home/kivikakk/.rbenv/versions/ruby-trunk/bin/ruby test.rb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7ffff7ff4700 (LWP 23118)]

Program received signal SIGSEGV, Segmentation fault.
0x00005555555c84ef in succ_index_lookup (sd=0xe0458d48e0458948, x=-454013225) at iseq.c:3089
3089            return imm_block_rank_get(sd->imm_part[i], j);
(gdb) bt 30
#0  0x00005555555c84ef in succ_index_lookup (sd=0xe0458d48e0458948, x=-454013225) at iseq.c:3089
#1  0x00005555555c2c3f in get_insn_info_succinct_bitvector (iseq=0x555555cd0b18, pos=428210124447829719) at is
eq.c:1522
#2  0x00005555555c2c7b in get_insn_info (iseq=0x555555cd0b18, pos=428210124447829719) at iseq.c:1530
#3  0x00005555555c2ca0 in rb_iseq_line_no (iseq=0x555555cd0b18, pos=428210124447829719) at iseq.c:1598
#4  0x000055555571a13e in calc_lineno (iseq=0x555555cd0b18, pc=0x8) at vm_backtrace.c:37
#5  0x000055555571c1ec in rb_profile_frames (start=0, limit=2048, buff=0x7ffff6740530 <_stackprof+176>, lines=
0x7ffff6744530 <_stackprof+16560>) at vm_backtrace.c:1295
#6  0x00007ffff653ee12 in stackprof_record_sample () at stackprof.c:484
#7  0x00007ffff653efe7 in stackprof_job_handler (data=0x0) at stackprof.c:539
#8  0x00007ffff653f0d3 in stackprof_newobj_handler (tpval=93825000082360, data=0x0) at stackprof.c:561
#9  0x000055555572086d in tp_call_trace (tpval=93825000082360, trace_arg=0x7fffffffb440) at vm_trace.c:1077
#10 0x000055555571ed6e in exec_hooks_body (ec=0x555555acf8c8, list=0x555555acf340, trace_arg=0x7fffffffb440) a
t vm_trace.c:272
#11 0x000055555571ee65 in exec_hooks_unprotected (ec=0x555555acf8c8, vm=0x555555acef20, list=0x555555acf340, t
race_arg=0x7fffffffb440) at vm_trace.c:301
#12 0x000055555571f0ab in rb_exec_event_hooks (trace_arg=0x7fffffffb440, pop_p=0) at vm_trace.c:345
#13 0x00005555555895cc in rb_exec_event_hook_orig (ec=0x555555acf8c8, flag=1048576, self=93825000082320, id=0,
 called_id=0, klass=0, data=93825000082120, pop_p=0) at vm_core.h:1778
#14 0x000055555558afb7 in gc_event_hook_body (ec=0x555555acf8c8, objspace=0x555555acf650, event=1048576, data=
93825000082120) at gc.c:1810
#15 0x000055555558b242 in newobj_slowpath (klass=93824998289320, flags=8199, v1=0, v2=0, v3=0, objspace=0x5555
55acf650, wb_protected=1) at gc.c:1924
#16 0x000055555558b294 in newobj_slowpath_wb_protected (klass=93824998289320, flags=8199, v1=0, v2=0, v3=0, ob
jspace=0x555555acf650) at gc.c:1934
#17 0x000055555558b3c9 in newobj_of (klass=93824998289320, flags=8199, v1=0, v2=0, v3=0, wb_protected=1) at gc
.c:1966
#18 0x000055555558b462 in rb_wb_protected_newobj_of (klass=93824998289320, flags=8199) at gc.c:1982
#19 0x0000555555727121 in ary_alloc (klass=93824998289320) at array.c:437
#20 0x00005555557271b4 in ary_new (klass=93824998289320, capa=1) at array.c:466
#21 0x00005555557274bb in rb_ary_tmp_new_from_values (klass=93824998289320, n=1, elts=0x7fffffffb700) at array
.c:514
#22 0x00005555557275ae in rb_ary_new_from_values (n=1, elts=0x7fffffffb700) at array.c:526
#23 0x000055555578bf1c in group_by_i (i=247, hash=93825000082280, argc=1, argv=0x7fffffffb868, blockarg=8) at
enum.c:894
#24 0x00005555557006cf in vm_yield_with_cfunc (ec=0x555555acf8c8, captured=0x7ffff6e53f10, self=93825000082320
, argc=1, argv=0x7fffffffb868, block_handler=0) at vm_insnhelper.c:2569
#25 0x0000555555714697 in invoke_block_from_c_bh (ec=0x555555acf8c8, block_handler=140737335607059, argc=1, ar
gv=0x7fffffffb868, passed_block_handler=0, cref=0x0, is_lambda=0,
    force_blockarg=0) at vm.c:1079
#26 0x00005555557147f7 in vm_yield (ec=0x555555acf8c8, argc=1, argv=0x7fffffffb868) at vm.c:1119
#27 0x000055555571050e in rb_yield_0 (argc=1, argv=0x7fffffffb868) at vm_eval.c:972
#28 0x000055555571052d in rb_yield_1 (val=247) at vm_eval.c:978
#29 0x000055555571055f in rb_yield (val=247) at vm_eval.c:988
(More stack frames follow...)
```

#5 - 06/27/2018 12:15 AM - shyouhei (Shyouhei Urabe)

- Related to Bug #14809: calc_lineno() returns wrong location added

#6 - 06/27/2018 12:16 AM - shyouhei (Shyouhei Urabe)

This issue and #14809 are caused by the same problem.

#7 - 06/27/2018 09:28 AM - shyouhei (Shyouhei Urabe)

- Status changed from Assigned to Closed

Applied in changeset trunk|r63763.

---

give up insn attr handles_frame

I introduced this mechanism in r62051 to speed things up. Later it was reported that the change causes problems. I searched for workarounds but nothing seemed appropriate. I hereby officially give it up. The idea to move ADD_PC around was a mistake.

Fixes [Bug #14809] and [Bug #14834].

Signed-off-by: Urabe, Shyouhei shyouhei@ruby-lang.org

Files

pc-treatment.diff	777 Bytes	06/08/2018	kivikakk (Ashe Connor)