# Ruby - Bug #17984

## [BUG] try to mark T_NONE object

06/14/2021 09:50 AM - byroot (Jean Boussier)

| | | | |
|---|---|---|---|
| **Status:** | Closed | | |
| **Priority:** | Normal | | |
| **Assignee:** | tenderlovemaking (Aaron Patterson) | | |
| **Target version:** | | | |
| **ruby -v:** | 3.1.0-dev | **Backport:** | 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN |

**Description**

About 1% of our CI jobs fail with this error.

If necessary I can try to find a repro, but these GC issues are hard to pin down because they require GC.stress which is very slow.

```
<OBJ_INFO:gc_mark_ptr@gc.c:6687> 0x00007f6717407ee8 [2 M   ] T_NONE
/tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/compile_cache/yaml.rb:117: [BUG] try to ma
rk T_NONE object
ruby 3.1.0dev (2021-06-12T19:33:15Z canary-fix 38bf83611e) [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0119 p:---- s:0680 e:000679 CFUNC  :fetch
c:0118 p:0174 s:0672 e:000671 METHOD /tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/compi
le_cache/yaml.rb:117
...

-- Ruby level backtrace information ----------------------------------------
...
/tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/compile_cache/yaml.rb:117:in `load_file'
/tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/compile_cache/yaml.rb:117:in `fetch'

-- C level backtrace information -------------------------------------------
/usr/local/bin/ruby(rb_print_backtrace+0x11) [0x5581a8716618] vm_dump.c:759
/usr/local/bin/ruby(rb_vm_bugreport) vm_dump.c:1041
/usr/local/bin/ruby(bug_report_end+0x0) [0x5581a8549aef] error.c:777
/usr/local/bin/ruby(rb_bug_without_die) error.c:777
/usr/local/bin/ruby(die+0x0) [0x5581a851857a] error.c:785
/usr/local/bin/ruby(rb_bug) error.c:787
/usr/local/bin/ruby(gc_mark_ptr+0x151) [0x5581a856f7b1] gc.c:6688
/usr/local/bin/ruby(each_insn_value+0x5) [0x5581a85a629d] iseq.c:327
/usr/local/bin/ruby(iseq_extract_values) iseq.c:178
/usr/local/bin/ruby(rb_iseq_each_value) iseq.c:246
/usr/local/bin/ruby(rb_iseq_mark) iseq.c:342
/usr/local/bin/ruby(gc_mark_imemo+0x60) [0x5581a8570720] gc.c:6802
/usr/local/bin/ruby(gc_mark_children) gc.c:6884
/usr/local/bin/ruby(gc_mark_stacked_objects+0x3d) [0x5581a8574a0d] gc.c:7068
/usr/local/bin/ruby(gc_mark_stacked_objects_incremental) gc.c:7102
/usr/local/bin/ruby(gc_marks_rest) gc.c:8084
/usr/local/bin/ruby(gc_rest+0x170) [0x5581a8574db0] gc.c:9003
/usr/local/bin/ruby(gc_rest+0x8) [0x5581a8572df8] gc.c:8875
/usr/local/bin/ruby(garbage_collect) gc.c:8864
/usr/local/bin/ruby(garbage_collect_with_gvl+0x83) [0x5581a8572f03] gc.c:9178
/usr/local/bin/ruby(objspace_malloc_fixup+0x17) [0x5581a8576341] gc.c:11282
/usr/local/bin/ruby(objspace_xmalloc0) gc.c:11353
/usr/local/bin/ruby(str_new0+0xda) [0x5581a868512a] string.c:797
/tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/bootsnap.so(bs_storage_to_output+0x0) [0x7
f67ecc403e3] bootsnap.c:525
/tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/bootsnap.so(fetch_cached_data) bootsnap.c:
527
/tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/bootsnap.so(bs_fetch) bootsnap.c:735
/tmp/bundle/ruby/3.1.0/gems/bootsnap-1.7.5/lib/bootsnap/bootsnap.so(bs_rb_fetch) bootsnap.c:362
/usr/local/bin/ruby(vm_call_cfunc_with_frame+0x10e) [0x5581a86ec9de] vm_insnhelper.c:2943
```

```
/usr/local/bin/ruby(vm_sendish+0x30e) [0x5581a86f847e] vm_insnhelper.c:4521
/usr/local/bin/ruby(vm_exec_core+0xf7) [0x5581a87036a7] insns.def:773
/usr/local/bin/ruby(rb_vm_exec+0xbc) [0x5581a86f8c9c] vm.c:2169
/usr/local/bin/ruby(load_iseq_eval+0xa) [0x5581a85af74a] load.c:594
/usr/local/bin/ruby(require_internal) load.c:1065
/usr/local/bin/ruby(rb_require_string+0x24) [0x5581a85afa1e] load.c:1142
/usr/local/bin/ruby(rb_f_require) load.c:838
....
```

### Associated revisions

**Revision 2599d1a8dff29a2376f36c8cc301839b454fc064 - 07/07/2021 12:48 AM - tenderlovemaking (Aaron Patterson)**

Store the dup'd CDHASH in the object list during IBF load

Since b2fc592c304 nothing was holding a reference to the dup'd CDHASH
during IBF loading. If a GC happened to run during IBF load then the
copied hash wouldn't have anything to keep it alive. We don't really
want to keep the originally loaded CDHASH hash, so this patch just
overwrites the original hash with the copied / modified hash.

[Bug #17984] [ruby-core:104259]

**Revision 2599d1a8dff29a2376f36c8cc301839b454fc064 - 07/07/2021 12:48 AM - tenderlovemaking (Aaron Patterson)**

Store the dup'd CDHASH in the object list during IBF load

Since b2fc592c304 nothing was holding a reference to the dup'd CDHASH
during IBF loading. If a GC happened to run during IBF load then the
copied hash wouldn't have anything to keep it alive. We don't really
want to keep the originally loaded CDHASH hash, so this patch just
overwrites the original hash with the copied / modified hash.

[Bug #17984] [ruby-core:104259]

**Revision 2599d1a8 - 07/07/2021 12:48 AM - tenderlovemaking (Aaron Patterson)**

Store the dup'd CDHASH in the object list during IBF load

Since b2fc592c304 nothing was holding a reference to the dup'd CDHASH
during IBF loading. If a GC happened to run during IBF load then the
copied hash wouldn't have anything to keep it alive. We don't really
want to keep the originally loaded CDHASH hash, so this patch just
overwrites the original hash with the copied / modified hash.

[Bug #17984] [ruby-core:104259]

### History

**#1 - 06/14/2021 09:17 PM - tenderlovemaking (Aaron Patterson)**

*- Assignee set to tenderlovemaking (Aaron Patterson)*

Can you send me a core file and I can take a look?

**#2 - 06/28/2021 12:10 PM - byroot (Jean Boussier)**

I didn't have the core dump and this issue only happens very rarely. I just caught it again today and this time I have the core dump. I forwarded it to @tenderlovemaking (Aaron Patterson).

**#3 - 07/06/2021 10:28 PM - tenderlovemaking (Aaron Patterson)**

*- File 0001-Store-the-dup-d-CDHASH-in-the-object-list-during-IBF.patch added*

I think I was finally able to track this down. The iseq marks references by walking the instructions looking for Ruby objects. The bug occurred here:

```
(lldb) f 6
frame #6: 0x000055886314df28 ruby`rb_iseq_mark at iseq.c:178
   175             {
   176               VALUE op = code[pos + op_no + 1];
   177               if (!SPECIAL_CONST_P(op)) {
-> 178                   VALUE newop = func(data, op);
   179                   if (newop != op) {
   180                       code[pos + op_no + 1] = newop;
```

```
   181                     }
(lldb) p code + pos
(VALUE *) $22 = 0x00007f0fd261e700
(lldb)
```

So the instruction living at 0x00007f0fd261e700 had a bad reference.

I was finally able to disassemble the iseq from the core file.  Here is the instruction (including some surrounding):

```
0x7f0fd261e6e8 0093 getlocal_WC_0( 7 )
0x7f0fd261e6f8 0095 dup
0x7f0fd261e700 0096 opt_case_dispatch( CDHASH (VALUE)0x7f0fdabbc3a8, 24 )
0x7f0fd261e718 0099 putobject( (VALUE)0x7f0fea201e70 )
0x7f0fd261e728 0101 topn( 1 )
```

0x7f0fd261e700 corresponds to an opt_case_dispatch instruction, and it's trying to mark the CDHASH.  Unfortunately the CDHASH reference has gone bad:

```
(lldb) rp 0x7f0fdabbc3a8
bits: [LM   ]
T_NONE: (RBasic) *$23 = (flags = 64, klass = 139706070975120)
(lldb)
```

It looks like b2fc592c3046e60fdfbb5692d52cc7cbf814b6d0 introduced a dup on the CDHASH when loading from IBF ( https://github.com/ruby/ruby/commit/b2fc592c3046e60fdfbb5692d52cc7cbf814b6d0#diff-a5ba41b51e3655f9f244362a616282b5119d3e15dd6c52ee999bbdfcc5b86a77R10779 ).  Nothing keeps that hash alive during the process of IBF loading, so if GC happens to run during an IBF load then this hash could be collected.

IBF loading keeps a temporary array of Ruby objects in order to maintain liveness during load.  We actually only care about the copied hash (not the original), so I made a patch that replaces the original hash in our temporary object list.

The patch is here: https://github.com/ruby/ruby/pull/4630

I've also attached it.  Thanks @xrxr for helping me track this down!

#### #4 - 07/07/2021 12:49 AM - tenderlovemaking (Aaron Patterson)

*- Status changed from Open to Closed*

Applied in changeset git|2599d1a8dff29a2376f36c8cc301839b454fc064.

---

Store the dup'd CDHASH in the object list during IBF load

Since b2fc592c304 nothing was holding a reference to the dup'd CDHASH
during IBF loading.  If a GC happened to run during IBF load then the
copied hash wouldn't have anything to keep it alive.  We don't really
want to keep the originally loaded CDHASH hash, so this patch just
overwrites the original hash with the copied / modified hash.

[Bug #17984] [ruby-core:104259]

### Files

| | | | |
|---|---|---|---|
| 0001-Store-the-dup-d-CDHASH-in-the-object-list-during-IBF.patch | 1.4 KB | 07/06/2021 | tenderlovemaking (Aaron Patterson) |