Ruby - Bug #21021

"try to mark T_NONE object" with 3.4.1

01/09/2025 03:43 PM - Benoit_Tigeot (Benoit Tigeot)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.4.1 (2024-12-25 revision 48d4efcb85) +YJIT +PRISM [x86_64-linux]	**Backport:**	3.1: DONTNEED, 3.2: DONTNEED, 3.3: DONTNEED, 3.4: DONE

Description

Hello

We upgraded to 3.4.1 yesterday but we are seeing crash since then.

```
/bundle/ruby/3.4.0/gems/activejob-7.2.2.1/lib/active_job/enqueuing.rb:93: [BUG] try to mark T_NONE object
```

I saw the other issue related to ffi gem https://bugs.ruby-lang.org/issues/20694

But in our case the C level backtrace information looks different.

https://gist.github.com/benoittgt/13507c2000281aa7740bc782adab68c5

We migrated this part of the code to parallel->concurrent-ruby and we do not see the error yet again but I am a little bit worried we could see the issue again.

Related issues:

Has duplicate Ruby - Bug #21087: "try to mark T_NONE object" error in Rage/Ac...	**Closed**
Has duplicate Ruby - Bug #21034: try to mark T_NONE object error after upgrad...	**Closed**

Associated revisions

Revision 58ccce60cf5f3268e7ef27942b75e78fe2d78e75 - 01/29/2025 04:54 AM - alanwu (Alan Wu)

YJIT: Initialize locals in ISeqs defined with ... (#12660)

- YJIT: Fix indentation [ci skip]

Fixes: cdf33ed5f37f9649c482c3ba1d245f0d80ac01ce

- YJIT: Initialize locals in ISeqs defined with ...

Previously, callers of forwardable ISeqs moved the stack pointer up
without writing to the stack. If there happens to be a stale value in
the area skipped over, it could crash due to "try to mark T_NONE". Also,
the uninitialized local variables were observable through binding.

Initialize the locals to nil.

[Bug #21021]

Revision 58ccce60cf5f3268e7ef27942b75e78fe2d78e75 - 01/29/2025 04:54 AM - alanwu (Alan Wu)

YJIT: Initialize locals in ISeqs defined with ... (#12660)

- YJIT: Fix indentation [ci skip]

Fixes: cdf33ed5f37f9649c482c3ba1d245f0d80ac01ce

- YJIT: Initialize locals in ISeqs defined with ...

Previously, callers of forwardable ISeqs moved the stack pointer up

without writing to the stack. If there happens to be a stale value in
the area skipped over, it could crash due to "try to mark T_NONE". Also,
the uninitialized local variables were observable through binding.

Initialize the locals to nil.

[Bug #21021]

Revision 58ccce60 - 01/29/2025 04:54 AM - alanwu (Alan Wu)

YJIT: Initialize locals in ISeqs defined with ... (#12660)

- YJIT: Fix indentation [ci skip]

Fixes: cdf33ed5f37f9649c482c3ba1d245f0d80ac01ce

- YJIT: Initialize locals in ISeqs defined with ...

Previously, callers of forwardable ISeqs moved the stack pointer up
without writing to the stack. If there happens to be a stale value in
the area skipped over, it could crash due to "try to mark T_NONE". Also,
the uninitialized local variables were observable through binding.

Initialize the locals to nil.

[Bug #21021]

Revision 706f1d0573f0b807bee4c0cc8937b8f5b9b24ebd - 02/14/2025 01:44 AM - alanwu (Alan Wu)

YJIT: Initialize locals in ISeqs defined with ...

Backport of GH-12660:

```
Previously, callers of forwardable ISeqs moved the stack pointer up
without writing to the stack. If there happens to be a stale value in
the area skipped over, it could crash due to "try to mark T_NONE". Also,
the uninitialized local variables were observable through `binding`.

Initialize the locals to nil.

[Bug #21021]
```

Revision 706f1d0573f0b807bee4c0cc8937b8f5b9b24ebd - 02/14/2025 01:44 AM - alanwu (Alan Wu)

YJIT: Initialize locals in ISeqs defined with ...

Backport of GH-12660:

```
Previously, callers of forwardable ISeqs moved the stack pointer up
without writing to the stack. If there happens to be a stale value in
the area skipped over, it could crash due to "try to mark T_NONE". Also,
the uninitialized local variables were observable through `binding`.

Initialize the locals to nil.

[Bug #21021]
```

Revision 706f1d05 - 02/14/2025 01:44 AM - alanwu (Alan Wu)

YJIT: Initialize locals in ISeqs defined with ...

Backport of GH-12660:

```
Previously, callers of forwardable ISeqs moved the stack pointer up
without writing to the stack. If there happens to be a stale value in
the area skipped over, it could crash due to "try to mark T_NONE". Also,
the uninitialized local variables were observable through `binding`.

Initialize the locals to nil.

[Bug #21021]
```

History

#1 - 01/09/2025 04:05 PM - Benoit_Tigeot (Benoit Tigeot)

Benoit_Tigeot (Benoit Tigeot) wrote:

> We migrated this part of the code to parallel->concurrent-ruby and we do not see the error yet again but I am a little bit worried we could see the issue again.

I was wrong. We still have the issue. Here is a new crash dump : https://gist.github.com/benoittgt/f0ad6476002b2a33c30070833e1d17c5

#2 - 01/09/2025 05:10 PM - Benoit_Tigeot (Benoit Tigeot)

Benoit_Tigeot (Benoit Tigeot) wrote in #note-1:

> I was wrong. We still have the issue. Here is a new crash dump : https://gist.github.com/benoittgt/f0ad6476002b2a33c30070833e1d17c5

Same with last psych update (it was present in crash dump but an old version).
https://gist.github.com/benoittgt/13507c2000281aa7740bc782adab68c5?permalink_comment_id=5380956#gistcomment-5380956

#3 - 01/09/2025 08:26 PM - tenderlovemaking (Aaron Patterson)

Are you able to get a core file or a backtrace from gdb? The bug is that some object has a T_NONE reference and is trying to mark that reference. We can't really tell what object has a broken reference without a core file (or possibly a gdb backtrace).

#4 - 01/09/2025 10:33 PM - alanwu (Alan Wu)

There seems to be a weakmap bug that's been around since at least November 2024 that could be responsible:
http://ci.rvm.jp/results/trunk-O0@ruby-sp2-noble-docker/5392991

```
rb_obj_info_dump: @)◆◆
/tmp/ruby/src/trunk-O0/test/ruby/test_weakkeymap.rb:142: [BUG] try to mark T_NONE object
ruby 3.4.0dev (2024-11-05T22:08:35Z master 4203c70dfa) +PRISM [x86_64-linux]
-- Control frame information -----------------------------------------------
c:0018 p:---- s:0114 e:000113 CFUNC  :new
c:0017 p:0004 s:0110 e:000109 BLOCK  /tmp/ruby/src/trunk-O0/test/ruby/test_weakkeymap.rb:142
```

Latest occurrence from 2 days ago: http://ci.rvm.jp/results/trunk-yjit@ruby-sp2-noble-docker/5513233

#5 - 01/10/2025 08:52 AM - Benoit_Tigeot (Benoit Tigeot)

Thanks for your answers.

tenderlovemaking (Aaron Patterson) wrote in #note-3:

> Are you able to get a core file or a backtrace from gdb? The bug is that some object has a T_NONE reference and is trying to mark that reference. We can't really tell what object has a broken reference without a core file (or possibly a gdb backtrace).

I'm gonna try but it will take some time.

#6 - 01/10/2025 11:40 AM - Benoit_Tigeot (Benoit Tigeot)

We are not seeing the issue if we disable YJIT, but it could be a side effect.

#7 - 01/15/2025 04:41 PM - Benoit_Tigeot (Benoit Tigeot)

Sorry for the delay. I removed the concurrency mecanism and let our crontask ran multiple times. The crash output seems to be more interesting.

https://gist.github.com/benoittgt/13507c2000281aa7740bc782adab68c5?permalink_comment_id=5391753#gistcomment-5391753

```
/bundle/ruby/3.4.0/gems/psych-5.2.2/lib/psych.so(parse+0x5c5) [0x7f3274e2bbd5] /bundle/ruby/3.4.0/gems/psych-5
.2.2/ext/psych/psych_parser.c:384
[0x7f326bd3b3cf]
```

#8 - 01/15/2025 05:30 PM - tenderlovemaking (Aaron Patterson)

Odd. This may be a weak map bug as @alanwu (Alan Wu) is saying.

The C level back trace has these lines:

```
/usr/local/lib/libruby.so.3.4(rb_gc_mark_vm_stack_values) /usr/include/ruby-3.4.1/gc.c:2346
/usr/local/lib/libruby.so.3.4(rb_execution_context_mark+0x39) [0x7f329134af49] /usr/include/ruby-3.4.1/vm.c:34
15
```

The GC is scanning the VM stack marking any Ruby objects it finds in the stack. This means something has pushed an invalid reference on the Ruby stack.

Do you know if any of the code in your Ruby level backtrace are using WeakMaps?

#9 - 01/15/2025 07:58 PM - alanwu (Alan Wu)

T_NONE on the stack is reminiscent of a class of YJIT bugs we see during development. I recommend building Ruby while passing --enable-yjit=dev to ./configure then attempting to re-trigger the crash. This build configuration runs debug assertions that can reveal more information about the bug. Note that you'll need cargo for this development build configuration and the build process will download some Rust dependencies from the internet.

If you use a third-party tool to build Ruby, you'll need to pass options to ./configure through that tool.

- For ruby-install, it's $ ruby-install -- --enable-yjit=dev
- For ruby-build, you can use the CONFIGURE_OPTS environment variable, e.g $ CONFIGURE_OPTS=--enable-yjit=dev ruby-build ....

You should be able to verify that you have a dev build by checking $ ruby --yjit -v. It should include "+YJIT dev" like the following:

```
ruby 3.4.1 (2024-12-25 revision 48d4efcb85) +YJIT dev +PRISM [arm64-darwin24]
```

#10 - 01/15/2025 10:38 PM - Benoit_Tigeot (Benoit Tigeot)

tenderlovemaking (Aaron Patterson) wrote in #note-8:

> Do you know if any of the code in your Ruby level backtrace are using WeakMaps?

I see no matching between the two

```
~/.rbenv/versions/3.4.1/lib/ruby/gems/3.4.0/gems  rg WeakMap -g '*.rb' --max-count 1
debug-1.10.0/lib/debug/source_repository.rb
32:        @cmap = ObjectSpace::WeakMap.new

bundler-2.6.2/lib/bundler/vendor/connection_pool/lib/connection_pool.rb
49:    INSTANCES = ObjectSpace::WeakMap.new

connection_pool-2.5.0/lib/connection_pool.rb
49:    INSTANCES = ObjectSpace::WeakMap.new

activerecord-7.2.2.1/lib/active_record/connection_adapters/pool_config.rb
16:      INSTANCES = ObjectSpace::WeakMap.new

activerecord-7.2.2.1/lib/active_record/connection_adapters/abstract/transaction.rb
190:          @lazy_enrollment_records ||= ObjectSpace::WeakMap.new

mustermann-3.0.3/lib/mustermann/equality_map.rb
3:[Omitted long line with 1 matches]

sorbet-runtime-0.5.11751/lib/types/types/typed_array.rb
32:          ObjectSpace::WeakMap.new[1] = 1

sorbet-runtime-0.5.11751/lib/types/types/typed_class.rb
50:            ObjectSpace::WeakMap.new[1] = 1

sorbet-runtime-0.5.11751/lib/types/types/simple.rb
81:          ObjectSpace::WeakMap.new[1] = 1

activesupport-7.2.2.1/lib/active_support/descendants_tracker.rb
18:      # On MRI `ObjectSpace::WeakMap` keys are weak references.

drb-2.2.1/lib/drb/weakidconv.rb
17:        @map = ObjectSpace::WeakMap.new
```

Thanks Alan for the detailed guide. I was able to use YJIT dev, get a crash but the output seems to be quite similar at first sight. I have a valid version

```
$ ruby --yjit -v
ruby 3.4.1 (2024-12-25 revision 48d4efcb85) +YJIT dev +PRISM [x86_64-linux]
```

Here is a dump https://gist.github.com/benoittgt/74d83534b9a2d8837d643cdcad318367

I've look a little bit before but those are mostly app logs. I'm gonna looked a little bit at yjit source code to see what can be look at.

I saw that someone posted a core file https://bugs.ruby-lang.org/issues/21034

Thanks

#11 - 01/16/2025 04:37 PM - Benoit_Tigeot (Benoit Tigeot)

Benoit_Tigeot (Benoit Tigeot) wrote in #note-10:

> I've look a little bit before but those are mostly app logs.

I am wondering if we could provide much more info on the crash dump. Could it be possible?

#12 - 01/16/2025 07:52 PM - tenderlovemaking (Aaron Patterson)

Benoit_Tigeot (Benoit Tigeot) wrote in #note-11:

> Benoit_Tigeot (Benoit Tigeot) wrote in #note-10:
>
> > I've look a little bit before but those are mostly app logs.
>
> I am wondering if we could provide much more info on the crash dump. Could it be possible?

We could add more info, but the problem is that the crash dump is happening too late. Something pushed a T_NONE on the VM stack, and by the time it crashes it's pretty hard to tell *who* did it. It might be possible to find with a core file, but would take some digging.

If you're able to build from source and reproduce the problem, could you try applying this patch:

```
diff --git a/tool/ruby_vm/views/_insn_entry.erb b/tool/ruby_vm/views/_insn_entry.erb
index 6ec33461c4..bc9a1d44b4 100644
--- a/tool/ruby_vm/views/_insn_entry.erb
+++ b/tool/ruby_vm/views/_insn_entry.erb
@@ -64,7 +64,7 @@ INSN_ENTRY(<%= insn.name %>)
     INC_SP(INSN_ATTR(sp_inc));
  %   insn.rets.reverse_each.with_index do |ret, i|
     TOPN(<%= i %>) = <%= insn.cast_to_VALUE ret %>;
-    VM_ASSERT(!RB_TYPE_P(TOPN(<%= i %>), T_NONE));
+    assert(!RB_TYPE_P(TOPN(<%= i %>), T_NONE));
     VM_ASSERT(!RB_TYPE_P(TOPN(<%= i %>), T_MOVED));
  %   end
  % end
```

The above patch should crash the process when something pushes a T_NONE on the stack. If we can catch the problem at the time it gets pushed, we can probably figure out why it's happening.

#13 - 01/17/2025 01:20 PM - Benoit_Tigeot (Benoit Tigeot)

tenderlovemaking (Aaron Patterson) wrote in #note-12:

> If you're able to build from source and reproduce the problem, could you try applying this patch:

Thanks Aaron. I was able to make a custom docker image with this patch proposal but for the moment I am not able to reproduce any error. I will let the cron task run and see if I can catch an occurence of crash.

#14 - 01/17/2025 05:22 PM - tenderlovemaking (Aaron Patterson)

Benoit_Tigeot (Benoit Tigeot) wrote in #note-13:

> tenderlovemaking (Aaron Patterson) wrote in #note-12:
>
> > If you're able to build from source and reproduce the problem, could you try applying this patch:
>
> Thanks Aaron. I was able to make a custom docker image with this patch proposal but for the moment I am not able to reproduce any error. I will let the cron task run and see if I can catch an occurence of crash.

Sounds good, thanks.

If this doesn't catch the error *before* GC, and it only reproduces with YJIT, then I suspect a YJIT bug. AFAIK, the only way anything can get pushed on the VM stack is either via VM instructions, or via machine code generated from YJIT.

#15 - 01/18/2025 11:45 AM - Benoit_Tigeot (Benoit Tigeot)

tenderlovemaking (Aaron Patterson) wrote in #note-14:

> If this doesn't catch the error *before* GC, and it only reproduces with YJIT, then I suspect a YJIT bug. AFAIK, the only way anything can get pushed on the VM stack is either via VM instructions, or via machine code generated from YJIT.

Ok, I got a crash again with the custom Ruby build with the assert but as you mention, it was not cached before. Alan do you have any idea of what can I do? I can make easily custom build, so I could add more logs maybe. https://gist.github.com/benoittgt/7a9ad5223ab2587827f09bc8ef9144b3

I look a little bit to implement a similar assert in yjit code but I didn't succeed. I also looked at previous additions, like https://github.com/ruby/ruby/commit/9d9865d9bc2f563c2c600ec53cc71926441b973f.

#16 - 01/23/2025 09:20 AM - alanwu (Alan Wu)

The following patch combined with the crash output should narrow the culprit down to a couple entries in the stack trace. It prints an index of the "Ruby level backtrace information" section, but it relies on a heuristic so could be off by a small amount.
You can then try putting GC.stress = true/false around a region of code mentioned in the stack trace to further narrow things down. It should up the crash rate, and if it does, that shows the bug is somewhere in the stress region.

```
diff --git a/gc.c b/gc.c
index 1ec159a2da..c4d79c222b 100644
--- a/gc.c
+++ b/gc.c
@@ -2342,7 +2342,14 @@ rb_gc_mark_values(long n, const VALUE *values)
 void
 rb_gc_mark_vm_stack_values(long n, const VALUE *values)
 {
+    int frame_flag_count = 0;
     for (long i = 0; i < n; i++) {
+        if (VM_FRAME_MAGIC_METHOD <= values[i] && values[i] < 0x80000000ul) { // see VM_FRAME_MAGIC_MASK
+            frame_flag_count++;
+        }
+        if (RB_TYPE_P(values[i], T_NONE)) {
+            rb_bug("T_NONE on stack. Seems to be from %dth frame", frame_flag_count);
+        }
         gc_mark_and_pin_internal(values[i]);
     }
 }
```

#17 - 01/23/2025 05:29 PM - Benoit_Tigeot (Benoit Tigeot)

Thanks Alan

The rb_bug was not printed. I followed something similar to Jean's work and enable GC.stress closer to the region. The code was very very slow with this mode. Here is a crash report https://gist.github.com/benoittgt/1e5b3054a8045f261cde21d0de25bb4a

#18 - 01/23/2025 06:05 PM - byroot (Jean Boussier)

```
/home/appuser/app/models/kubernetes_item.rb: [BUG] Segmentation fault at 0x000055bf92769c2c
ruby 3.4.1 (2024-12-25 revision 48d4efcb85) +YJIT dev +PRISM [x86_64-linux]

-- Control frame information -----------------------------------------------
c:0071 p:---- s:0418 e:000417 DUMMY  [FINISH]
c:0070 p:---- s:0415 e:000414 CFUNC  :compile_file
```

Looks like it crash while compiling your app/models/kubernetes_item.rb. Are you able to reproduce this consistently?

If so, if you could share that file it would be ideal, but if you can't then one way to narrow this down more could be to "bisect" the file by removing parts of it until it stops failing, to try to figure out the pattern in the code that cause the crash.

#19 - 01/23/2025 07:11 PM - alanwu (Alan Wu)

Here's some "bisecting" pointers to narrow this down some more. Are you able to crash it without bootsnap? Specifically without the iseq cache, but you should be able to turn off everything with the DISABLE_BOOTSNAP env var. Also, it's crashing during/after a Kernel#require, what's the file being required at the time of crash? Is it "/home/appuser/app/models/kubernetes_cluster.rb"? You can also try enabling GC.stress for just the body of the filed required.

The dummy frame on top is unusual. It might be there for arbitrarily inserting a line of backtrace, but no such line seems to be there.

#20 - 01/23/2025 09:19 PM - alanwu (Alan Wu)

- Related to Bug #21087: "try to mark T_NONE object" error in Rage/ActiveRecord/Fiber with 3.4.1 upgrade added

#21 - 01/24/2025 12:11 PM - Benoit_Tigeot (Benoit Tigeot)

I started again with a fresh build with the last [two patch proposals[(https://github.com/ruby/ruby/compare/master...benoittgt:ruby:crash-on-t_none). Yjit is enable in dev mode, bootsnap too (ruby 3.4.1 (2025-01-23 revision e4722fe585) +YJIT dev +PRISM [x86_64-linux]). Our build process is complicated so I started from scratch I think the last crash report is interesting. It reports Alan patch proposal.
https://gist.github.com/benoittgt/141abcef14a78b97e637c677bae12eca

byroot (Jean Boussier) wrote in #note-18:

> Looks like it crash while compiling your app/models/kubernetes_item.rb. Are you able to reproduce this consistently?

That's the main issue. It's random. Sometimes is on every cron task every 30min, sometimes once for few hours. I added logs to try to narrow the issue but similar logs are repeated thousands of time before the crash. kubernetes_item.rb use psych and YAML.load(content, permitted_classes: [Time, Symbol, Date]). I am wondering if this could be the issue, for example for a very specific YAML. If I increase concurency with Concurrent::FixedThreadPool.new(20) and Concurrent::Future.execute(executor: pool) do it seems I can increase the chance of crash.

alanwu (Alan Wu) wrote in #note-19:

> Are you able to crash it without bootsnap?

I'm gonna try now.

alanwu (Alan Wu) wrote in #note-19:

> You can also try enabling GC.stress for just the body of the filed required.

Using GC.stress complelty block the app. Locally I'm getting quickly 100% CPU usage from the Ruby process.

#22 - 01/24/2025 12:17 PM - byroot (Jean Boussier)

> Using GC.stress completely block the app.

Yes, GC.stress isn't really viable except for short snippets.

I suspect we've seen this bug in some form at Shopify too. It's likely some missing write barrier in the ISeq loading code, or something similar. But that's terribly hard to track down :/

#23 - 01/24/2025 01:11 PM - Benoit_Tigeot (Benoit Tigeot)

byroot (Jean Boussier) wrote in #note-22:

> But that's terribly hard to track down :/

Arf.
The "good" news, is I am able to reproduce at 100% if I use concurrency.

alanwu (Alan Wu) wrote in #note-19:

> Are you able to crash it without bootsnap?

I see no positiv impact without bootsnap. I see no longer reference in the crash report but I still have /bundle/ruby/3.4.0/gems/psych-5.2.3/lib/psych/tree_builder.rb:97: [BUG] T_NONE on stack. Seems to be from 67th frame

#24 - 01/24/2025 04:39 PM - travisbell (Travis Bell)

Benoit_Tigeot (Benoit Tigeot) wrote in #note-23:

> The "good" news, is I am able to reproduce at 100% if I use concurrency.

For what it's worth on #21034, that environment is a Falcon environment so yes, concurrency is in play there as well. What's interesting is that a different service of ours which is also using Falcon, never crashes. So simply "being concurrent" isn't enough to trigger it, although I believe this appears to be something all of these tickets have in common, YJIT + concurrency.

#25 - 01/28/2025 02:41 AM - hsbt (Hiroshi SHIBATA)

- Related to Bug #21034: try to mark T_NONE object error after upgrading to 3.4.1 added

#26 - 01/29/2025 12:57 AM - alanwu (Alan Wu)

I have a patch that should address the issue. It definitely fixes *a* bug, seems to fix the workload in #21087. If you'd like, try it on for size and share if it works!

```
diff --git a/yjit/src/codegen.rs b/yjit/src/codegen.rs
```

```
index 37ddbce0bb..e8153c17b0 100644
--- a/yjit/src/codegen.rs
+++ b/yjit/src/codegen.rs
@@ -8068,7 +8068,6 @@ fn gen_send_iseq(
         }
     }

-    // Don't nil fill forwarding iseqs
     if !forwarding {
         // Nil-initialize missing optional parameters
         nil_fill(
@@ -8103,9 +8102,13 @@ fn gen_send_iseq(
         assert_eq!(1, num_params);
         // Write the CI in to the stack and ensure that it actually gets
         // flushed to memory
+        asm_comment!(asm, "put call info for forwarding");
         let ci_opnd = asm.stack_opnd(-1);
         asm.ctx.dealloc_reg(ci_opnd.reg_opnd());
         asm.mov(ci_opnd, VALUE(ci as usize).into());
+
+        // Nil-initialize other locals which are above the CI
+        nil_fill("nil-initialize locals", 1..num_locals, asm);
     }

     // Points to the receiver operand on the stack unless a captured environment is used
```

(This is https://github.com/ruby/ruby/pull/12660 modulo tests so it applies onto 3.4.1 cleanly.)

#27 - 01/29/2025 05:49 AM - alanwu (Alan Wu)

- Status changed from Open to Closed

Applied in changeset git|58ccce60cf5f3268e7ef27942b75e78fe2d78e75.

YJIT: Initialize locals in ISeqs defined with ... (#12660)

- YJIT: Fix indentation [ci skip]

Fixes: cdf33ed5f37f9649c482c3ba1d245f0d80ac01ce

- YJIT: Initialize locals in ISeqs defined with ...

Previously, callers of forwardable ISeqs moved the stack pointer up
without writing to the stack. If there happens to be a stale value in
the area skipped over, it could crash due to "try to mark T_NONE". Also,
the uninitialized local variables were observable through binding.

Initialize the locals to nil.

[Bug #21021]

#28 - 01/29/2025 03:49 PM - Benoit_Tigeot (Benoit Tigeot)

alanwu (Alan Wu) wrote in #note-26:

> If you'd like, try it on for size and share if it works!

Running the 3.4.1 with your patch right now. No issues for the moment. Thanks a lot!

I will post an update tomorrow UTC, with more runs of the code impacted.

#29 - 01/30/2025 01:07 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: UNKNOWN to 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: REQUIRED

#30 - 01/30/2025 07:49 AM - Benoit_Tigeot (Benoit Tigeot)

Benoit_Tigeot (Benoit Tigeot) wrote in #note-28:

> I will post an update tomorrow UTC, with more runs of the code impacted.

I can confirm that Alan's patch is fixing our issue. No crash on our side even with some concurrency in the application code.

#31 - 01/30/2025 04:05 PM - alanwu (Alan Wu)

- Has duplicate Bug #21087: "try to mark T_NONE object" error in Rage/ActiveRecord/Fiber with 3.4.1 upgrade added

#32 - 01/30/2025 04:05 PM - alanwu (Alan Wu)

- Related to deleted (Bug #21087: "try to mark T_NONE object" error in Rage/ActiveRecord/Fiber with 3.4.1 upgrade)

#33 - 01/30/2025 04:06 PM - alanwu (Alan Wu)

- Has duplicate Bug #21034: try to mark T_NONE object error after upgrading to 3.4.1 added

#34 - 01/30/2025 04:06 PM - alanwu (Alan Wu)

- Related to deleted (Bug #21034: try to mark T_NONE object error after upgrading to 3.4.1)

#35 - 01/30/2025 04:12 PM - alanwu (Alan Wu)

- Backport changed from 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: REQUIRED to 3.1: DONTNEED, 3.2: DONTNEED, 3.3: DONTNEED, 3.4: REQUIRED

#36 - 02/14/2025 01:45 AM - k0kubun (Takashi Kokubun)

- Backport changed from 3.1: DONTNEED, 3.2: DONTNEED, 3.3: DONTNEED, 3.4: REQUIRED to 3.1: DONTNEED, 3.2: DONTNEED, 3.3: DONTNEED, 3.4: DONE

ruby_3_4 706f1d0573f0b807bee4c0cc8937b8f5b9b24ebd.