Ruby - Bug #20679

Rails CI errors since abc04e898b627ab37fa9dd5e330f239768778d8b

08/16/2024 11:08 PM - yahonda (Yasuo Honda)

Status:	Closed		
Priority:	Normal		
Assignee:	jeremyevans0 (Jeremy Evans)		
Target version:			
ruby -v:	ruby 3.4.0dev (2024-08-15T18:33:13Z :detached: 2c6e16eb51) [x86_64-linux]	**Backport:**	3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN

Description

Rails CI against Ruby master branch gets failed since abc04e898b627ab37fa9dd5e330f239768778d8b

Steps to reproduce

```
git clone https://github.com/rails/rails
cd rails/railties
bundle install
bin/test test/application/rendering_test.rb:39
```

Expected behavior

It should pass as 2c6e16eb51 does.

```
$ ruby -v
ruby 3.4.0dev (2024-08-15T18:33:13Z :detached: 2c6e16eb51) [x86_64-linux]
$ bin/test test/application/rendering_test.rb:39
/home/yahonda/.rbenv/versions/trunk/lib/ruby/gems/3.4.0+0/gems/json-2.7.1/lib/json/common.rb:3: wa
rning: ostruct was loaded from the standard library, but will no longer be part of the default gem
s starting from Ruby 3.5.0.
You can add ostruct to your Gemfile or gemspec to silence this warning.
/home/yahonda/.rbenv/versions/trunk/lib/ruby/gems/3.4.0+0/gems/json-2.7.1/lib/json/common.rb:3: wa
rning: ostruct was loaded from the standard library, but will no longer be part of the default gem
s starting from Ruby 3.5.0.
You can add ostruct to your Gemfile or gemspec to silence this warning.
Run options: --seed 50444

# Running:

.

Finished in 1.554218s, 0.6434 runs/s, 1.2868 assertions/s.
1 runs, 2 assertions, 0 failures, 0 errors, 0 skips
$
```

Actual behavior

It gets NoMethodError: undefined method '[]' for nil.

```
$ ruby -v
ruby 3.4.0dev (2024-08-15T20:00:09Z :detached: abc04e898b) [x86_64-linux]
$ bin/test test/application/rendering_test.rb:39
/home/yahonda/.rbenv/versions/trunk/lib/ruby/gems/3.4.0+0/gems/json-2.7.1/lib/json/common.rb:3: wa
rning: ostruct was loaded from the standard library, but will no longer be part of the default gem
s starting from Ruby 3.5.0.
You can add ostruct to your Gemfile or gemspec to silence this warning.
/home/yahonda/.rbenv/versions/trunk/lib/ruby/gems/3.4.0+0/gems/json-2.7.1/lib/json/common.rb:3: wa
rning: ostruct was loaded from the standard library, but will no longer be part of the default gem
s starting from Ruby 3.5.0.
You can add ostruct to your Gemfile or gemspec to silence this warning.
Run options: --seed 53551
```

```
# Running:

E

Error:
ApplicationTests::RenderingTest#test_Unknown_format_falls_back_to_HTML_template:
NoMethodError: undefined method '[]' for nil
    /home/yahonda/.rbenv/versions/trunk/lib/ruby/gems/3.4.0+0/gems/rack-test-2.1.0/lib/rack/test.r
b:275:in 'Rack::Test::Session#parse_uri'
    /home/yahonda/.rbenv/versions/trunk/lib/ruby/gems/3.4.0+0/gems/rack-test-2.1.0/lib/rack/test.r
b:161:in 'Rack::Test::Session#custom_request'
    /home/yahonda/.rbenv/versions/trunk/lib/ruby/gems/3.4.0+0/gems/rack-test-2.1.0/lib/rack/test.r
b:112:in 'Rack::Test::Session#get'
    /home/yahonda/.rbenv/versions/trunk/lib/ruby/3.4.0+0/forwardable.rb:240:in 'Rack::Test::Method
s#get'
    test/application/rendering_test.rb:39:in 'block in <class:RenderingTest>'


bin/test test/application/rendering_test.rb:19



Finished in 1.499265s, 0.6670 runs/s, 0.0000 assertions/s.
1 runs, 0 assertions, 0 failures, 1 errors, 0 skips
$
```

Associated revisions

Revision ea7ceff82cec98d0c419e9807dcb33dcc08b56fa - 08/20/2024 02:00 AM - jeremyevans (Jeremy Evans)

Avoid hash allocation for certain proc calls

Previously, proc calls such as:

```
proc{|| }.(**empty_hash)
proc{|b: 1| }.(**r2k_array_with_empty_hash)
```

both allocated hashes unnecessarily, due to two separate code paths.

The first call goes through CALLER_SETUP_ARG/vm_caller_setup_keyword_hash,
and is simple to fix by not duping an empty keyword hash that will be
dropped.

The second case is more involved, in setup_parameters_complex, but is
fixed the exact same way as when the ruby2_keywords hash is not empty,
by flattening the rest array to the VM stack, ignoring the last
element (the empty keyword splat). Add a flatten_rest_array static
function to handle this case.

Update test_allocation.rb to automatically convert the method call
allocation tests to proc allocation tests, at least for the calls
that can be converted. With the code changes, all proc call
allocation tests pass, showing that proc calls and method calls
now allocate the same number of objects.

I've audited the allocation tests, and I believe that all of the low
hanging fruit has been collected. All remaining allocations are
either caller side:

- Positional splat + post argument
- Multiple positional splats
- Literal keywords + keyword splat
- Multiple keyword splats

Or callee side:

- Positional splat parameter
- Keyword splat parameter
- Keyword to positional argument conversion for methods that don't accept keywords
- ruby2_keywords method called with keywords

Reapplies abc04e898b627ab37fa9dd5e330f239768778d8b, which was reverted at
d56470a27c5a8a2e7aee7a76cea445c2d29c0c59, with the addition of a bug fix and
test.

Fixes [Bug #20679]

Revision ea7ceff82cec98d0c419e9807dcb33dcc08b56fa - 08/20/2024 02:00 AM - jeremyevans (Jeremy Evans)

Avoid hash allocation for certain proc calls

Previously, proc calls such as:

```
proc{|| }.(**empty_hash)
proc{|b: 1| }.(**r2k_array_with_empty_hash)
```

both allocated hashes unnecessarily, due to two separate code paths.

The first call goes through CALLER_SETUP_ARG/vm_caller_setup_keyword_hash,
and is simple to fix by not duping an empty keyword hash that will be
dropped.

The second case is more involved, in setup_parameters_complex, but is
fixed the exact same way as when the ruby2_keywords hash is not empty,
by flattening the rest array to the VM stack, ignoring the last
element (the empty keyword splat). Add a flatten_rest_array static
function to handle this case.

Update test_allocation.rb to automatically convert the method call
allocation tests to proc allocation tests, at least for the calls
that can be converted. With the code changes, all proc call
allocation tests pass, showing that proc calls and method calls
now allocate the same number of objects.

I've audited the allocation tests, and I believe that all of the low
hanging fruit has been collected. All remaining allocations are
either caller side:

- Positional splat + post argument
- Multiple positional splats
- Literal keywords + keyword splat
- Multiple keyword splats

Or callee side:

- Positional splat parameter
- Keyword splat parameter
- Keyword to positional argument conversion for methods that don't accept keywords
- ruby2_keywords method called with keywords

Reapplies abc04e898b627ab37fa9dd5e330f239768778d8b, which was reverted at
d56470a27c5a8a2e7aee7a76cea445c2d29c0c59, with the addition of a bug fix and
test.

Fixes [Bug #20679]

Revision ea7ceff8 - 08/20/2024 02:00 AM - jeremyevans (Jeremy Evans)

Avoid hash allocation for certain proc calls

Previously, proc calls such as:

```
proc{|| }.(**empty_hash)
proc{|b: 1| }.(**r2k_array_with_empty_hash)
```

both allocated hashes unnecessarily, due to two separate code paths.

The first call goes through CALLER_SETUP_ARG/vm_caller_setup_keyword_hash,
and is simple to fix by not duping an empty keyword hash that will be
dropped.

The second case is more involved, in setup_parameters_complex, but is
fixed the exact same way as when the ruby2_keywords hash is not empty,
by flattening the rest array to the VM stack, ignoring the last
element (the empty keyword splat). Add a flatten_rest_array static
function to handle this case.

Update test_allocation.rb to automatically convert the method call
allocation tests to proc allocation tests, at least for the calls
that can be converted. With the code changes, all proc call
allocation tests pass, showing that proc calls and method calls
now allocate the same number of objects.

I've audited the allocation tests, and I believe that all of the low
hanging fruit has been collected. All remaining allocations are
either caller side:

- Positional splat + post argument
- Multiple positional splats
- Literal keywords + keyword splat
- Multiple keyword splats

Or callee side:

- Positional splat parameter
- Keyword splat parameter
- Keyword to positional argument conversion for methods that don't accept keywords
- ruby2_keywords method called with keywords

Reapplies abc04e898b627ab37fa9dd5e330f239768778d8b, which was reverted at
d56470a27c5a8a2e7aee7a76cea445c2d29c0c59, with the addition of a bug fix and
test.

Fixes [Bug #20679]

History

#1 - 08/16/2024 11:22 PM - k0kubun (Takashi Kokubun)

- Assignee set to jeremyevans0 (Jeremy Evans)

#2 - 08/16/2024 11:46 PM - jeremyevans0 (Jeremy Evans)

Thanks for the report. I will revert and then try to work on a fix.

#3 - 08/17/2024 12:59 AM - jeremyevans0 (Jeremy Evans)

Reverted in d56470a27c5a8a2e7aee7a76cea445c2d29c0c59

Here's a failing example that does not require Rails:

```
class A
   def self.get(uri, params = {}, env = {}, &block)
     env['HTTPS']
   end

  ruby2_keywords def get(*args, &block)
    A.get(*args, &block)
  end
end

A.new.get("/pages/foo", {}, "HTTPS" => "on")
```

Definitely related to the ruby2_keywords optimization. I'll keep digging to find the root cause. I'll leave this open until I have a working fix.

Rails hits this through use of forwardable in rack-test. We should also consider switching forwardable to use generic argument forwarding (...) instead of ruby2_keywords when using Ruby 2.7+. It's simpler, and faster on Ruby 3.4 as generic argument forwarding is now allocationless.

#4 - 08/19/2024 10:09 PM - jeremyevans0 (Jeremy Evans)

jeremyevans0 (Jeremy Evans) wrote in #note-3:

> Definitely related to the ruby2_keywords optimization. I'll keep digging to find the root cause. I'll leave this open until I have a working fix.

I submitted a pull request for the fix: https://github.com/ruby/ruby/pull/11409

Root cause was when I extracted a function, I missed the use of a local variable after the extraction. This wasn't caught because the same variable existed in a higher scope.

> Rails hits this through use of forwardable in rack-test. We should also consider switching forwardable to use generic argument forwarding (...) instead of ruby2_keywords when using Ruby 2.7+. It's simpler, and faster on Ruby 3.4 as generic argument forwarding is now allocationless.

I submitted a pull request to change forwardable to use generic argument forwarding: https://github.com/ruby/forwardable/pull/34

#5 - 08/20/2024 02:00 AM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|ea7ceff82cec98d0c419e9807dcb33dcc08b56fa.

Avoid hash allocation for certain proc calls

Previously, proc calls such as:

```
proc{|| }.(**empty_hash)
proc{|b: 1| }.(**r2k_array_with_empty_hash)
```

both allocated hashes unnecessarily, due to two separate code paths.

The first call goes through CALLER_SETUP_ARG/vm_caller_setup_keyword_hash,
and is simple to fix by not duping an empty keyword hash that will be
dropped.

The second case is more involved, in setup_parameters_complex, but is
fixed the exact same way as when the ruby2_keywords hash is not empty,
by flattening the rest array to the VM stack, ignoring the last
element (the empty keyword splat). Add a flatten_rest_array static
function to handle this case.

Update test_allocation.rb to automatically convert the method call
allocation tests to proc allocation tests, at least for the calls
that can be converted. With the code changes, all proc call
allocation tests pass, showing that proc calls and method calls
now allocate the same number of objects.

I've audited the allocation tests, and I believe that all of the low
hanging fruit has been collected. All remaining allocations are
either caller side:

- Positional splat + post argument
- Multiple positional splats
- Literal keywords + keyword splat
- Multiple keyword splats

Or callee side:

- Positional splat parameter
- Keyword splat parameter
- Keyword to positional argument conversion for methods that don't accept keywords
- ruby2_keywords method called with keywords

Reapplies abc04e898b627ab37fa9dd5e330f239768778d8b, which was reverted at
d56470a27c5a8a2e7aee7a76cea445c2d29c0c59, with the addition of a bug fix and
test.

Fixes [Bug #20679]