Ruby master - Feature #16103
Make the dot-colon method reference frozen
08/14/2019 12:23 PM - maciej.mensfeld (Maciej Mensfeld)

Status: Closed
Priority: Normal
Assignee: ko1 (Koichi Sasada)
Target version:

Description
I made a PR to freeze the dot-colon method reference result object (https://github.com/ruby/ruby/pull/2267). Nobu asked to make an issue out of that. I initially discussed that with Matz and Ko1 during the hack challenge in Bristol.

Here are some of the reasons why I think it should be done before releasing this feature:

- It's worth keeping the bound methods frozen as one should not modify them in general
- Freezing keeps a window of opportunity for introducing method reference caching in case it would be needed because the method object is immutable.
- I want to work on the "last method" reference cache for that feature when we see more use-cases after 2.7 is released and without having it frozen, the cost and complexity are probably higher than the outcome.

Example of the misuse that makes GC life much harder (note, that freezing does not fix that, but allows further work related to this issue):

```ruby
# frozen_string_literal: true

GC.disable

class Parse
  def self.call(string)
    string.to_f
  end
end

class Normalize
  def self.call(number)
    number.round
  end
end

class Transform
  def self.call(int)
    int * 2
  end
end

# Simulate a long running data producing source with batch results
stream = Array.new(10_000) { Array.new(100) { '100.2' } }

before = GC.stat[:total_allocated_objects]

# This will provide batches to be processed, let's assume an endless data-source
stream.each do |batch|
  batch
    .map{&Parse::call}
    .map{&Normalize::call}
    .map{&Transform::call}
end

after = GC.stat[:total_allocated_objects]
p a = after - before # 150001
before = GC.stat[:total_allocated_objects]
```
```
# "cache"
parse = Parse.call
normalize = Normalize.call
transform = Transform.call

stream.each do |batch|
    batch
        .map(&parse)
        .map(&normalize)
        .map(&transform)
end

after = GC.stat[:total_allocated_objects]
p b = after - before # 120004

p "Difference: #{a - b}" # "Difference: 29997"

Things get even more "problematic" when referencing like above is not used on batches but on each of the objects separately:

# frozen_string_literal: true
GC.disable

class Parse
    def self.call(string)
        string.to_f
    end
end

class Normalize
    def self.call(number)
        number.round
    end
end

class Transform
    def self.call(int)
        int * 2
    end
end

# Simulate a long running data producing source with batch results
stream = Array.new(10_000) { Array.new(100) { '100.2' } }

before = GC.stat[:total_allocated_objects]

# This will provide batches to be processed, let's assume an endless data-source
stream.each do |batch|
    batch.map do |message|
        message
            .then(&Parse.call)
            .then(&Normalize.call)
            .then(&Transform.call)
    end
end

after = GC.stat[:total_allocated_objects]

p a = after - before # 12010002

before = GC.stat[:total_allocated_objects]

# This will provide batches to be processed, let's assume an endless data-source
parse = Parse.call
normalize = Normalize.call
```
transform = Transform.call

stream.each do |batch|
  batch
    .map(&:parse)
    .map(&:normalize)
    .map(&:transform)
end

after = GC.stat[:total_allocated_objects]

p b = after - before # 12004
p "Difference: #{a - b}" # "Difference: 11889998"

I am aware, that this won't help in case we don't reuse the same object method references multiple times but based on many use-cases from here [https://bugs.ruby-lang.org/issues/13581](https://bugs.ruby-lang.org/issues/13581) and my own when building data-intense applications, it's not uncommon to build "pipelines" of things applied to each of the batches as a set of functions.

Apart from that I would also vote on freezing the .method(:method) result object as well, but I know Matz was against.

Associated revisions

Revision abe12d8b - 08/29/2019 07:58 AM - nobu (Nobuyoshi Nakada)
Freeze method reference operator object
[Feature #16103]

Revision c45dd4d4 - 08/30/2019 09:24 AM - maciej.mensfeld (Maciej Mensfeld)
Make the dot-colon method reference frozen
[Feature #16103]
Close: [https://github.com/ruby/ruby/pull/2267](https://github.com/ruby/ruby/pull/2267)

History

#1 - 08/29/2019 07:37 AM - ko1 (Koichi Sasada)
Matz accepted it :tada:

#2 - 08/29/2019 07:37 AM - ko1 (Koichi Sasada)
- Assignee set to ko1 (Koichi Sasada)

#3 - 08/29/2019 08:05 AM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Closed

Applied in changeset gitiabe12d8b96be3c12618811f22ca01798366f99f8.

Freeze method reference operator object
[Feature #16103]

#4 - 11/05/2019 09:16 PM - Eregon (Benoit Daloze)
FWIW, I think it would be better to freeze all Method and UnboundMethod objects. Is there any use-case for adding instance variables to them?