## Ruby master - Bug #18919

### Ractor: can't share #Method objects

07/15/2022 10:07 PM - chuke (Tiago Cardoso)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td></td>
</tr>
<tr>
<td>Backport:</td>
<td>2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN</td>
</tr>
</tbody>
</table>

**Description**

The following is not shareable:

```ruby
meth = ::Kernel.method(:BigDecimal)
=> #<Method: Kernel.BigDecimal(*)> <internal:ractor>:816:in `make_shareable': can not make shareable object for #<Method: Kernel.BigDecimal(*)> (Ractor::Error)
```

I understand that procs have the issue of accessing outer-scope variables, but does the same apply to methods converted to procs?

### History

**#1 - 07/20/2022 03:18 AM - shan (Shannon Skipper)**

This is just an aside, but for what it's worth you can pass the class and method name and reconstitute a Method instance inside a Ractor.

```ruby
require 'bigdecimal'

meth = Kernel.method(:BigDecimal)

ractor = Ractor.new(meth.owner, meth.name) do |meth_owner, meth_name|
  meth_owner.method(meth_name)
end

ractor.take.call(42)
#=> 0.42e2
```

**#2 - 07/20/2022 10:51 AM - Eregon (Benoit Daloze)**

I don't see a .to_proc call in the description. Either way, both Method and Proc objects can't be shared, they have mutable state.

For Method/UnboundMethod they could potentially be made immutable and therefore shareable (sounds good to me), but it would be a potentially-breaking change.

**#3 - 07/26/2022 08:30 AM - chuke (Tiago Cardoso)**

- Subject changed from Ractor: can't share method-to-proc objects to Ractor: can't share #Method objects

Indeed, I didn't mean "to-proc", Method objects, as per the usage example, is the one I meant.