Ruby - Feature #17291

Optimize __send__ call

10/29/2020 03:05 AM - mrkn (Kenta Murata)

Status:	Assigned
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

I made a patch to optimize a __send__ call. This optimization replaces a __send__ method call with a call of the method whose name is the first argument of __send__ method. The patch is available in this pull-request.

By this change, the redefined __send__ method is no longer called when it is called by a symbol method name. I guess it is no problem because the following warning message is displayed for a long time.

```
$ ruby -e 'def __send__; end'
-e:1: warning: redefining `__send__' may cause serious problems
```

This proposal introduces two new instructions: sendsym and opt_sendsym_without_block. These instructions handle the cases that the first argument of __send__ method is not a symbol literal. I think I can combine these two instructions into one if prefered.

This proposal includes the change proposed in #17288. I'll mark it as a duplicate of this proposal.

I don't handle send method in this proposal. The reason is that we need to examine the redefinition of send method in the instruction execution time. I want to discuss only __send__ method in this ticket.

The benchmark result is below:

```
# Iteration per second (i/s)

|                 |compare-ruby|built-ruby|
|:----------------|-----------:|---------:|
|vm_send_sym      |     18.001M|  112.208M|
|                 |           -|     6.23x|
|vm_send_var      |     17.779M|   30.922M|
|                 |           -|     1.74x|
|vm_send_var_alt  |      3.817M|    6.817M|
|                 |           -|     1.79x|
```

Related issues:

Has duplicate Ruby - Feature #17288: Optimize __send__ call with a literal me...	**Assigned**

History

#1 - 10/29/2020 03:06 AM - mrkn (Kenta Murata)

- Has duplicate Feature #17288: Optimize __send__ call with a literal method name added

#2 - 10/30/2020 12:21 AM - shyouhei (Shyouhei Urabe)

I'm neutral (at least no against it). __send__ in general has other usages than to reroute method visibilities. Optimising it could benefit good wills.

#3 - 11/04/2020 12:17 AM - mrkn (Kenta Murata)

I found that rspec-core redefines __send__.

https://github.com/rspec/rspec-mocks/blob/461d7f6f7f688f69154e410dcd9c7690120e7dbf/lib/rspec/mocks/verifying_double.rb#L45-L53

#4 - 11/04/2020 03:29 AM - shyouhei (Shyouhei Urabe)

It seems this leaks memory?

```
`nproc --all`.to_i.times.map do |i|
  Ractor.new :"#{i}_0" do |sym|
    while true do
      __send__(sym = sym.succ) rescue nil
```

```
    end
  end
end

while true do
  sleep 1
  GC.start
  p GC.stat(:heap_live_slots)
end
```

I see very different output comparing the proposed implementation versus master.

#5 - 11/04/2020 10:53 AM - Eregon (Benoit Daloze)

Yeah I don't think a warning is good enough to prevent people overriding it.
It should be an exception if the goal is to prevent overriding it.

I think we should not compromise semantics for optimizations, that usually leads to more complicated semantics that alternative Ruby implementations have to replicate (which is nonsense if those implementations would check it correctly if redefined).
As an example, the optimization for Hash#each_pair led to very confusing semantics where lambdas/Method#to_proc appear to unsplat/destructure arguments (they do not, it's just a side effect of the incorrect optimization).

#6 - 11/06/2020 04:45 PM - mrkn (Kenta Murata)

shyouhei (Shyouhei Urabe) wrote in #note-4:

> It seems this leaks memory?
>
> ```
> `nproc --all`.to_i.times.map do |i|
>   Ractor.new :"#{i}_0" do |sym|
>     while true do
>       __send__(sym = sym.succ) rescue nil
>     end
>   end
> end
>
> while true do
>   sleep 1
>   GC.start
>   p GC.stat(:heap_live_slots)
> end
> ```
>
> I see very different output comparing the proposed implementation versus master.

I used SYM2ID in compile_call function and sendsym and opt_sendsym_without_block instructions. This SYM2ID makes dynamic symbols permanent, so many symbols remained in the heap. This is the reason for the observed phenomenon.

I added a commit to fix this bug.

#7 - 11/06/2020 04:45 PM - mrkn (Kenta Murata)

- Status changed from Open to Assigned

#8 - 11/06/2020 04:46 PM - mrkn (Kenta Murata)

The new benchmark result is below:

```
# Iteration per second (i/s)

|                     |compare-ruby|built-ruby|
|:--------------------|-----------:|---------:|
|vm_send_sym          |     18.265M|  113.593M|
|                     |           -|     6.22x|
|vm_send_var          |     17.750M|   31.974M|
|                     |           -|     1.80x|
|vm_send_var_alt      |      3.955M|    7.499M|
|                     |           -|     1.90x|
|vm_send_sym_missing  |      7.135M|    8.982M|
|                     |           -|     1.26x|
|vm_send_var_missing  |      7.271M|    7.454M|
|                     |           -|     1.03x|
```

#9 - 01/12/2021 05:47 AM - mrkn (Kenta Murata)

mrkn (Kenta Murata) wrote in #note-3:

> I found that rspec-core redefines __send__.
>
> https://github.com/rspec/rspec-mocks/blob/461d7f6f7f688f69154e410dcd9c7690120e7dbf/lib/rspec/mocks/verifying_double.rb#L45-L53

This redefinition is used to distinguish the form of the method call in a mock object.
rspec-mocks recognizes that the form of recv.__send__(:meth) is used when __send__ is called before method_missing is called, or the form of recv.meth is used when otherwise.

The feature to distinguish method calling form is necessary to keep the compatibility if we employ this optimization of __send__ call.