Speedup `block.call` where `block` is passed block parameter.

01/07/2018 04:58 PM - ko1 (Koichi Sasada)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>ko1 (Koichi Sasada)</td>
</tr>
<tr>
<td>Target version</td>
<td>2.6</td>
</tr>
</tbody>
</table>

**Description**

**abstract**

Speedup block.call where block is passed block parameter with a special object named "proxyblock" which responds to call and invoke passed block.

**background**

Ruby 2.5 improved performance of passing a passed block parameter by lazy Proc creation ([Feature #14045](https://www.ruby-lang.org/en/stdlib-2.5.0/libdoc/lambda/rdoc/Proc.html)). However, block.call (block is a passed block parameter) is not optimized and need to create a Proc object immediately.

**proposal**

We need to make Proc creation lazily for performance. There are several way to achieve it, but I propose to use special object named "blockproxy" object.

This is a pseudo code to use it:

```
# block is given block parameter
block.call(1, 2, 3)
 #=> translate at compile time

if block is not modified and
   block is ISeq/IFunc block
   tmp = blockproxy
else
   tmp = block # create Proc and so on
end

tmp.call(1, 2, 3)
```

blockproxy.call invoke given block if Proc#call is not redefined, otherwise make a Proc and call Proc#call as usual.

Advantage of this method is we can also use this technique with the safe navigation operator (block&.call).

If block is not given, then tmp will be nil, and no method dispatched with &..

Note that this technique depends on the assumption "we can't access to the evaluated receiver just before method dispatch". We don't/can't access blockproxy object at method dispatch, and no compatibility issue.

**evaluation**

Using [https://github.com/k0kubun/benchmark_driver](https://github.com/k0kubun/benchmark_driver)

```
prelude: |
  def block_yield
    yield
  end
  def bp_yield &b
    yield
  end
  def bp_call &b
```
```ruby
b.call
end

def bp_safe_call &b
  b&.call
end

benchmark:
  - block_yield{}
  - bp_yield{}
  - bp_call{}
  - bp_safe_call{}
  - bp_safe_call

Result:

Warming up --------------------------------------

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>block_yield{}</td>
<td>1.298M i/100ms</td>
</tr>
<tr>
<td>bp_yield{}</td>
<td>1.177M i/100ms</td>
</tr>
<tr>
<td>bp_call{}</td>
<td>447.723k i/100ms</td>
</tr>
<tr>
<td>bp_safe_call{}</td>
<td>413.261k i/100ms</td>
</tr>
<tr>
<td>bp_safe_call</td>
<td>2.955M i/100ms</td>
</tr>
</tbody>
</table>

Calculating -------------------------------------

<table>
<thead>
<tr>
<th>Method</th>
<th>Value</th>
<th>Value</th>
<th>Time</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>trunk</td>
<td>20.672M</td>
<td>20.808M</td>
<td>51.933M</td>
<td>2.512265s</td>
</tr>
<tr>
<td>block_yield{}</td>
<td>16.294M</td>
<td>16.220M</td>
<td>47.099M</td>
<td>2.890459s</td>
</tr>
<tr>
<td>bp_yield{}</td>
<td>5.626M</td>
<td>14.752M</td>
<td>17.909M</td>
<td>3.182966s</td>
</tr>
<tr>
<td>bp_call{}</td>
<td>5.555M</td>
<td>14.557M</td>
<td>16.530M</td>
<td>2.975892s</td>
</tr>
<tr>
<td>bp_safe_call{}</td>
<td>31.339M</td>
<td>23.561M</td>
<td>118.184M</td>
<td>3.771157s</td>
</tr>
</tbody>
</table>

The patch is here:
https://gist.github.com/ko1/d8a1a9d92075b27a8e95ca528cc57fd2

Related issues:
Related to Ruby master - Bug #14335; block.call should respect redefinition o...
Closed

Associated revisions

Revision 7fd11834 - 01/07/2018 07:18 PM - ko1 (Koichi Sasada)
Speedup block.call [Feature #14330]

- insns.def (getblockparamproxy): introduce new instruction to return
  the rb_block_param_proxy object if possible. This object responds
  to call method and invoke given block (completely similar to yield).

- method.h (OPTIMIZED_METHOD_TYPE_BLOCK_CALL): add new optimized call type
  which is for rb_block_param_proxy.cal.

- vm_insnhelper.c (vm_call_method_each_type): ditto.

- vm_insnhelper.c (vm_call_opt_block_call): ditto.

- vm_core.h (BOP_CALL, PROC_REDEFINED_OP_FLAG): add check for Proc#call
  redefinition.

- compile.c (iseq_compile_each0): compile to use new insn
  getblockparamproxy for method call.

```
- vm_insnhelper.c (vm_call_method_each_type): ditto.
- vm_insnhelper.c (vm_call_opt_block_call): ditto.
- vm_core.h (BOP_CALL, PROC_REDEFINED_OP_FLAG): add check for Proc#call redefinition.
- compile.c (iseq_compile_each0): compile to use new insn getblockparamproxy for method call.

Revision 61659 - 01/07/2018 07:18 PM - ko1 (Koichi Sasada)
Speedup block.call [Feature #14330]

- insns.def (getblockparamproxy): introduce new instruction to return the rb_block_param_proxy object if possible. This object responds to call method and invoke given block (completely similar to yield).
- method.h (OPTIMIZED_METHOD_TYPE_BLOCK_CALL): add new optimized call type which is for rb_block_param_proxy.cal.
- vm_insnhelper.c (vm_call_method_each_type): ditto.
- vm_insnhelper.c (vm_call_opt_block_call): ditto.
- vm_core.h (BOP_CALL, PROC_REDEFINED_OP_FLAG): add check for Proc#call redefinition.
- compile.c (iseq_compile_each0): compile to use new insn getblockparamproxy for method call.

Revision 633b4638 - 02/21/2018 08:14 AM - ko1 (Koichi Sasada)
add NEWS entries about [Feature #14318] and [Feature #14330].
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@62514 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 62514 - 02/21/2018 08:14 AM - ko1 (Koichi Sasada)
add NEWS entries about [Feature #14318] and [Feature #14330].

Revision 62514 - 02/21/2018 08:14 AM - ko1 (Koichi Sasada)
add NEWS entries about [Feature #14318] and [Feature #14330].

History
#1 - 01/07/2018 07:18 PM - ko1 (Koichi Sasada)
- Status changed from Open to Closed

Applied in changeset trunk@61659.

Speedup block.call [Feature #14330]

- insns.def (getblockparamproxy): introduce new instruction to return the rb_block_param_proxy object if possible. This object responds to call method and invoke given block (completely similar to yield).
- method.h (OPTIMIZED_METHOD_TYPE_BLOCK_CALL): add new optimized call type which is for rb_block_param_proxy.cal.
- vm_insnhelper.c (vm_call_method_each_type): ditto.
- vm_insnhelper.c (vm_call_opt_block_call): ditto.
- vm_core.h (BOP_CALL, PROC_REDEFINED_OP_FLAG): add check for Proc#call redefinition.
- compile.c (iseq_compile_each0): compile to use new insn getblockparamproxy for method call.
getblockparamproxy for method call.

#2 - 01/08/2018 09:00 AM - nobu (Nobuyoshi Nakada)
- Related to Bug #14335: block.call should respect redefinition of Proc#call added