Ruby master - Feature #15902

Add a specialized instruction for `.nil?`

06/05/2019 04:28 PM - tenderlovemaking (Aaron Patterson)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

I'd like to add a specialized instruction for .nil?. We have specialized instructions for .length and .empty?, and surprisingly our application also calls .nil? a lot:

```
[aaron@TC ~/g/github (gc-boot-stats)]$ git grep '.empty?' | wc -l
    2553
[aaron@TC ~/g/github (gc-boot-stats)]$ git grep '.length' | wc -l
    3975
[aaron@TC ~/g/github (gc-boot-stats)]$ git grep '.nil?' | wc -l
    3117
```

I'm not sure how hot any of the .nil? callsites are, but I think this instruction will speed up most of them.

I tried two benchmark runners:

Benchmark/ips

```
require "benchmark/ips"

class Niller
  def nil?; true; end
end

not_nil = Object.new
xnil = nil
niller = Niller.new

Benchmark.ips do |x|
  x.report("nil?")    { xnil.nil? }
  x.report("not nil") { not_nil.nil? }
  x.report("niller")   { niller.nil? }
end
```

Results

On Ruby master:

```
[aaron@TC ~/g/ruby (master)]$ ./ruby compil.rb
Warming up --------------------------------------
                nil?   429.195k i/100ms
             not nil   437.889k i/100ms
              niller   437.935k i/100ms
Calculating -------------------------------------
                nil?     20.166M (± 8.1%) i/s -    100.002M in   5.002794s
             not nil     20.046M (± 7.6%) i/s -     99.839M in   5.020086s
              niller     22.467M (± 6.1%) i/s -    112.111M in   5.013817s
[aaron@TC ~/g/ruby (master)]$ ./ruby compil.rb
Warming up --------------------------------------
                nil?   449.660k i/100ms
             not nil   433.836k i/100ms
              niller   443.073k i/100ms
Calculating -------------------------------------
                nil?     19.997M (± 8.8%) i/s -     99.375M in   5.020458s
             not nil     20.529M (± 7.0%) i/s -    102.385M in   5.020689s
```

```
              niller     21.796M (± 8.0%) i/s -    108.110M in   5.002300s
[aaron@TC ~/g/ruby (master)]$ ./ruby compil.rb
Warming up --------------------------------------
                 nil?   402.119k i/100ms
             not nil   438.968k i/100ms
              niller   398.226k i/100ms
Calculating -------------------------------------
                 nil?     20.050M (±12.2%) i/s -     98.519M in   5.008817s
             not nil     20.614M (± 8.0%) i/s -    102.280M in   5.004531s
              niller     22.223M (± 8.8%) i/s -    110.309M in   5.013106s
```

On this patch:

```
[aaron@TC ~/g/ruby (specialized-nilp)]$ ./ruby compil.rb
Warming up --------------------------------------
                 nil?   468.371k i/100ms
             not nil   456.517k i/100ms
              niller   454.981k i/100ms
Calculating -------------------------------------
                 nil?     27.849M (± 7.8%) i/s -    138.169M in   5.001730s
             not nil     26.417M (± 8.7%) i/s -    131.020M in   5.011674s
              niller     21.561M (± 7.5%) i/s -    107.376M in   5.018113s
[aaron@TC ~/g/ruby (specialized-nilp)]$ ./ruby compil.rb
Warming up --------------------------------------
                 nil?   477.259k i/100ms
             not nil   428.712k i/100ms
              niller   446.109k i/100ms
Calculating -------------------------------------
                 nil?     28.071M (± 7.3%) i/s -    139.837M in   5.016590s
             not nil     25.789M (±12.9%) i/s -    126.470M in   5.011144s
              niller     20.002M (±12.2%) i/s -     98.144M in   5.001737s
[aaron@TC ~/g/ruby (specialized-nilp)]$ ./ruby compil.rb
Warming up --------------------------------------
                 nil?   467.676k i/100ms
             not nil   445.791k i/100ms
              niller   415.024k i/100ms
Calculating -------------------------------------
                 nil?     26.907M (± 8.0%) i/s -    133.755M in   5.013915s
             not nil     25.319M (± 7.9%) i/s -    125.713M in   5.007758s
              niller     19.569M (±11.8%) i/s -     96.286M in   5.008533s
```

According to benchmark/ips, it's about 27% faster when the object is nil or a regular object. When it's an object that implements .nil?, I think it might be slower but it's hard to tell.

Benchmark-driver

I added a benchmark driver file:

```
prelude: |
  class Niller; def nil?; true; end; end
  xnil, notnil = nil, Object.new
  niller = Niller.new
benchmark:
  - xnil.nil?
  - notnil.nil?
  - niller.nil?
loop_count: 10000000
```

Results (tested against master @ c9b74f9fd95113df903fc34cc1d6ec3fb3160c85)

```
[aaron@TC ~/g/ruby (specialized-nilp)]$ make benchmark ARGS=benchmark/nil_p.yml
./revision.h unchanged
/Users/aaron/.rbenv/shims/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/benchmark-d
river/exe/benchmark-driver \
                --executables="compare-ruby::/Users/aaron/.rbenv/shims/ruby --disable=gems -I.ext/
common --disable-gem" \
```

```
                --executables="built-ruby::./miniruby -I./lib -I. -I.ext/common  ./tool/runruby.rb
 --extout=.ext  -- --disable-gems --disable-gem" \
                benchmark/nil_p.yml
Calculating -------------------------------------
                     compare-ruby  built-ruby
           xnil.nil?      68.825M    405.121M i/s -     10.000M times in 0.145296s 0.024684s
         notnil.nil?      66.357M    267.874M i/s -     10.000M times in 0.150700s 0.037331s
         niller.nil?     110.273M    123.089M i/s -     10.000M times in 0.090684s 0.081242s

Comparison:
                        xnil.nil?
          built-ruby: 405120725.0 i/s
        compare-ruby:  68825019.2 i/s - 5.89x  slower

                       notnil.nil?
          built-ruby: 267873885.2 i/s
        compare-ruby:  66357000.6 i/s - 4.04x  slower

                       niller.nil?
          built-ruby: 123089042.6 i/s
        compare-ruby: 110273035.8 i/s - 1.12x  slower

[aaron@TC ~/g/ruby (specialized-nilp)]$ make benchmark ARGS=benchmark/nil_p.yml
./revision.h unchanged
/Users/aaron/.rbenv/shims/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/benchmark-d
river/exe/benchmark-driver \
                --executables="compare-ruby::/Users/aaron/.rbenv/shims/ruby --disable=gems -I.ext/
common --disable-gem" \
                --executables="built-ruby::./miniruby -I./lib -I. -I.ext/common  ./tool/runruby.rb
 --extout=.ext  -- --disable-gems --disable-gem" \
                benchmark/nil_p.yml
Calculating -------------------------------------
                     compare-ruby  built-ruby
           xnil.nil?      45.083M    360.998M i/s -     10.000M times in 0.221811s 0.027701s
         notnil.nil?      69.558M    271.054M i/s -     10.000M times in 0.143765s 0.036893s
         niller.nil?     115.423M     79.667M i/s -     10.000M times in 0.086638s 0.125523s

Comparison:
                        xnil.nil?
          built-ruby: 360997801.1 i/s
        compare-ruby:  45083426.9 i/s - 8.01x  slower

                       notnil.nil?
          built-ruby: 271054130.3 i/s
        compare-ruby:  69557959.1 i/s - 3.90x  slower

                       niller.nil?
        compare-ruby: 115422793.6 i/s
          built-ruby:  79666674.5 i/s - 1.45x  slower
```

I think there is too much noise for the third case.

I'm not happy about making rb_false non-static, but I'm not sure how else to do this patch.

What do you think?

History

#1 - 06/05/2019 04:55 PM - Eregon (Benoit Daloze)

FWIW, we already inline nil? in the AST for TruffleRuby:
https://github.com/oracle/truffleruby/commit/57628008#diff-09677670ed37ee212c217374c6468718

Makes sense since it's such a small operation and it's frequently used.

#2 - 07/19/2019 05:09 PM - tenderlovemaking (Aaron Patterson)

I applied this optimization and tested against our application while logging instructions:

```
> insn_info[order(insn_info$Count),]
              Instruction     Count
94                reverse         1
95       opt_newarray_max        49
93          setblockparam        54
92     opt_call_c_function        60
22          opt_aset_with       302
78              setglobal       417
39       setclassvariable       441
87                opt_div       597
88                   once      1263
91              branchnil      1357
52       opt_regexpmatch1      1386
53                duphash      1513
76              getglobal      1538
58                   topn      1607
56                opt_and      2129
89                 opt_or      2344
86               opt_mult      2859
46              toregexp     11460
28          definesmethod     11880
20            setconstant     12930
81                 intern     19087
5              defineclass     32953
55                 opt_mod     33719
24          opt_aref_with     36994
80                  throw     44954
35              duparray     53448
96              setlocal     58770
30            getspecial     63765
69            opt_length     74235
77              newrange     77459
71              opt_size     98244
15           definemethod    104818
42              tostring    124320
70                opt_gt    130296
74           concatarray    133235
48       getclassvariable    143407
34         opt_str_freeze    192432
68       opt_regexpmatch2    272036
90          checkkeyword    279475
44          freezestring    290999
14       putspecialobject    338706
43          concatstrings    344539
18              putstring    435151
45          getblockparam    455764
64            expandarray    490262
41              checktype    541355
21                newhash    698166
63                opt_neq    764259
57                   swap   1146326
66                 opt_ge   1283820
73                 opt_le   1345993
51             opt_nil_p   1536827
72             opt_minus   1598463
67           invokeblock   1614973
79     getblockparamproxy   1675137
82              getlocal   1799493
38              opt_ltlt   1888635
65            splatarray   1971596
83      opt_case_dispatch   2049897
85          setlocal_WC_1   2062346
50            invokesuper   2153631
84                 opt_lt   2229708
3      opt_setinlinecache   2422345
54              opt_plus   2803688
60           opt_empty_p   2820474
27              opt_aset   3204087
47   putobject_INT2FIX_0_   3790567
75          adjuststack   3924393
31              opt_not   3986585
2             getconstant   4722342
62                  jump   4781094
49                  dupn   4854287
37    setinstancevariable   5385254
```

```
61             checkmatch   5448507
36               newarray   5539621
26                   setn   6319773
59   putobject_INT2FIX_1_   9551850
16                defined  10071630
32          getlocal_WC_1  15007475
29                 opt_eq  16058544
10                   send  16372975
9                putobject  20392124
4                   putnil  26095091
19               branchif  30425098
33               opt_aref  31976718
1      opt_getinlinecache  32302934
40    getinstancevariable  34616623
13                    pop  36429507
7           setlocal_WC_0  39844133
23                    dup  41106221
11                    nop  51714288
8                 putself  56476479
17           branchunless  62428811
12                  leave  86843816
6  opt_send_without_block 147141336
25          getlocal_WC_0 213198244
```

Our application boot process executes this instruction 1.5mil times:

```
> insn_info[which(insn_info$Instruction == "opt_nil_p"),]
   Instruction   Count
51   opt_nil_p 1536827
>
```

I think this is a useful instruction, so I'd like to apply the patch. :)

#3 - 07/30/2019 07:45 AM - ko1 (Koichi Sasada)

All right, go ahead.

#4 - 07/31/2019 11:26 PM - tenderlovemaking (Aaron Patterson)

- Status changed from Open to Closed

I pushed this as 9faef3113fb4331524b81ba73005ba13fa0ef6c6

We found it decreased our app boot time by ~1.8 seconds (which is not large percentage wise, but definitely measurable). I tested by booting our app in production mode 50 times with and without the patch.

Here is a graph of the results:

62253666-7da11900-b3ab-11e9-8c16-5b39b6a04d82.png
Again with Y at 0:

62253946-5d258e80-b3ac-11e9-852f-f6fd39e152c3.png

#5 - 08/27/2019 07:50 AM - ned (Ned Hadzo)

This was reverted, right? https://github.com/ruby/ruby/commit/a0980f2446c0db735b8ffeb37e241370c458a626

#6 - 08/27/2019 11:33 AM - mame (Yusuke Endoh)

ned (Ned Hadzo) wrote:

> This was reverted, right? https://github.com/ruby/ruby/commit/a0980f2446c0db735b8ffeb37e241370c458a626

It has been re-introduced. The Mojave issue was also fixed.

Files

0001-Add-a-specialized-instruction-for-.nil-calls.patch	7.13 KB	06/05/2019	tenderlovemaking (Aaron Patterson)