Ruby master - Feature #15380

faster method lookup for Array#all? #none? #one?

12/05/2018 04:26 AM - fursich (Onishi Koji)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

summary

This PR proposes Array-specific implementations for #all? , #none? and #one? to enable faster method lookup.

Before this patch Array#all? was not implemented in Array class, so alternatively, Enumerable#all? was used each time the method is called.

On the other hand, #any? has its own method entry in Array class for faster method calls.

This patch provides above three methods with Array-specific implementations that are equivalent to what Array#any? has.

https://github.com/ruby/ruby/pull/2041

benchmark

```
**********************************************************************************
benchmarking Array#all?
**********************************************************************************

Calculating -------------------------------------
    Array#all? (new)          421.298 i/s -      1.000k times in 2.373616s (2.37ms/i)
    Array#all? (old)          335.364 i/s -      1.000k times in 2.981838s (2.98ms/i)

Comparison:
    Array#all? (new):        421.3 i/s
    Array#all? (old):        335.4 i/s - 1.26x  slower

<running with --jit>
Calculating -------------------------------------
    Array#all? (new)                244.929 i/s -      1.000k times in 4.082823s (4.08ms/i)
    Array#all? (old)                210.354 i/s -      1.000k times in 4.753895s (4.75ms/i)

Comparison:
    Array#all? (new):        244.9 i/s
    Array#all? (old):        210.4 i/s - 1.16x  slower
```

Attached benchmark shows the full benchmark results:

https://gist.github.com/fursich/1d1bad353ddc2f4b510b34e3191fd302

Each method gets approx. 10-20% faster with repeated calls.
It only impacts on method lookup (not execution itself), but at least it should makeArray#all? work as just efficiently as Array#any? does.

estimate of impact

Just to provide a rough picture on how frequently these methods are used in real world app, here shows a quick-and-dirty investigations I did with rails (using its latest master as of Dec 5):

```
rails (master)$ git grep '\.all?' | wc -l
      80
rails (master)$ git grep '\.one?' | wc -l
```

```
      13
rails (master)$ git grep '\.none?' | wc -l
      25
```

while

```
rails (master)$ git grep '\.any?' | wc -l
```

(* the result includes non-Array method. the intention here is just to give rough estimate on how frequently these methods are used compared with each other)

It's probably fair to say the use of the three methods (118 lines in total here) are *not* ignorably rare compared to #all?

motivation behind it

In developing Ruby apps we encounter (often non essential) discussion around 'which method call is faster?', 'should we use this method for efficiency?'.

As Ruby lover I really hope to pick methods based on pure readability and Ruby-ness,

Hopefully it helps Ruby become faster even at slightest level :)

Associated revisions

Revision 66212 - 12/05/2018 04:25 AM - nobu (Nobuyoshi Nakada)

implement Array-specific #all?, #none?, #one?

Before this patch Array#all? was not implemented in Array class
and alternatively Enumerable#all? was used, while #any? has its
own method entry in Array class. Similarly, Array#none? and #one?
also lacks its own implementation.
This patch provides Array-specific implementations for above three
methods to enable faster method lookup.

[Fix GH-2041]

From: Koji Onishi fursich0@gmail.com

History

#1 - 12/05/2018 04:55 AM - ko1 (Koichi Sasada)

If we introduce special versions for Array, some new code are introduced.
It means the code we need to maintain will be increased.

I think about 20% improvement is not worth compare with the disadvantage.

BTW, I don't check the patch and benchmark code, the length of Arrays affect the results?
If you want to show the benchmark results, pls consider such conditions.

Thanks,
Koichi

#2 - 12/05/2018 06:23 AM - shevegen (Robert A. Heiler)

While I agree with Koichi in regards to benchmarks and more code
meaning more maintenance work (naturally), I think it should still
be considered for the trade off being potentially worth it. If there
is time for discussion at the next developer meeting perhaps matz
could be asked.

#3 - 12/05/2018 10:11 PM - Hanmac (Hans Mackowiak)

nobu (Nobuyoshi Nakada) can you close this?