Ruby - Feature #12481

Add Array#view to allow opt-in copy-on-write sharing of Array contents

06/11/2016 03:50 AM - headius (Charles Nutter)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

Currently the only way to get a copy-on-write view of an existing array is to construct a new array, via #[] or partition or similar methods. I propose a new way to use an existing array to get a view of another array: Array#view (name negotiable).

```
ary1 = [1,2,3,4]
ary2 = Array.new(0)

ary2.view(ary1, 0, 2) # => ary2 now holds [2, 3] without any allocation.
```

This would be an OPTIONAL feature for VMs to implement, if they have copy-on-write array capabilities. Otherwise, it would just do the allocation necessary to hold a copy of the results.

Because of copy-on-write, there would be no visible connection between ary2 and ary1 above; any modifications to either would force a copy.

Consider the case of processing a large array of values N at a time; you could ary2.view in a tight loop, passing that to other functions, and never allocate a single object.

Official documentation would say something like "Make this array contain the contents of the target array starting at 'begin' for 'len' elements." COW would not be mentioned, but VMs could improve performance by using COW under the covers.

A similar feature String#view would also be useful for traversing a stream of bytes without doing any allocation. Think parsing.

History

#1 - 07/19/2016 07:27 AM - mrkn (Kenta Murata)

I expect array's views keep to refer the original array even if it is changed because it is "view".
This recalls RDB's view.

#2 - 07/19/2016 09:08 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

I first thought the view method is to create a view array with CoW. But this attaches an array as a view to another array. This allows reducing object allocation, but its side-effect can cause issues. I am slightly against the idea.

Matz.