Ruby trunk - Feature #6638

Array as queue

06/24/2012 04:41 PM - funny_falcon (Yura Sokolov)

Status:	Closed	
Priority:	Normal	
Assignee:	nobu (Nobuyoshi Nakada)	
Target version:	2.0.0	

Description

Many libraries use Array as queue (cause stdlib has no real dequeue class).
And typical pattern is to use #push and #shift methods with small count of #push
with following small count of #shift.

But currently big array makes a copy every time array is modified after #shift,
so that it degrades greatly when Array size growths.

Ironically, usage of #unshift with following #pop degrades much less, but most
libraries doesn't consider this fact, and other ruby's implementations suffers
from such pattern.

Test for 1.9.3 before patch: https://gist.github.com/2981959#file_test_before_patch

Main point of this patch is to change rb_ary_modify so that it considers ARY_SHARED_NUM
and steel shared array pointer when ARY_SHARED_NUM == 1.
To make it possible, it saves array's capa instead of array's length in ary_make_shared
and fixes rb_ary_sort_bang accordantly.

Test for 1.9.3 after patch: https://gist.github.com/2981959#file_test_after_patch
(note that gain is less for ruby-trunk but still exists)

Pull request https://github.com/ruby/ruby/pull/133
Patch https://github.com/ruby/ruby/pull/133.patch

(but I think, despite the patch, Ruby needs real Deque in stdlib)

Associated revisions

Revision 06de286c - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37581 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37581 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.

Revision 37581 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.

Revision 37581 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.

Revision 37581 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.

Revision 37581 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.

Revision 37581 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.

Revision b11975df - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: make array really suitable for queue

- array.c (ary_ensure_room_for_push): make array really suitable for queue. [Feature #6638] when array is shared (which happens after Array#shift), and ARY_SHARED_NUM == 1 (which is very often when array used as queue), then make rb_ary_push push directly into shared array.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37582 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37582 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: make array really suitable for queue

- array.c (ary_ensure_room_for_push): make array really suitable for queue. [Feature #6638] when array is shared (which happens after Array#shift), and ARY_SHARED_NUM == 1 (which is very often when array used as queue), then make rb_ary_push push directly into shared array.

Revision 37582 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: make array really suitable for queue

- array.c (ary_ensure_room_for_push): make array really suitable for queue. [Feature #6638] when array is shared (which happens after Array#shift), and ARY_SHARED_NUM == 1 (which is very often when array used as queue), then make rb_ary_push push directly into shared array.

Revision 37582 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: make array really suitable for queue

- array.c (ary_ensure_room_for_push): make array really suitable for queue. [Feature #6638] when array is shared (which happens after Array#shift), and ARY_SHARED_NUM == 1 (which is very often when array used as queue), then make rb_ary_push push directly into shared array.

Revision 37582 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: make array really suitable for queue

- array.c (ary_ensure_room_for_push): make array really suitable for queue. [Feature #6638] when array is shared (which happens after Array#shift), and ARY_SHARED_NUM == 1 (which is very often when array used as queue), then make rb_ary_push push directly into shared

array.

Revision 37582 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: make array really suitable for queue

- array.c (ary_ensure_room_for_push): make array really suitable for queue. [Feature #6638] when array is shared (which happens after Array#shift), and ARY_SHARED_NUM == 1 (which is very often when array used as queue), then make rb_ary_push push directly into shared array.

Revision 37582 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: make array really suitable for queue

- array.c (ary_ensure_room_for_push): make array really suitable for queue. [Feature #6638] when array is shared (which happens after Array#shift), and ARY_SHARED_NUM == 1 (which is very often when array used as queue), then make rb_ary_push push directly into shared array.

Revision aaa9cb1a - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: use shared array in rb_ary_slice

- array.c (rb_ary_splice): use shared array in rb_ary_slice. [Feature #6638]
  - use ary_ensure_room_for_push when rb_ary_slice used to add at the end of array, cause rb_ary_concat use rb_ary_slice.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37583 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37583 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: use shared array in rb_ary_slice

- array.c (rb_ary_splice): use shared array in rb_ary_slice. [Feature #6638]
  - use ary_ensure_room_for_push when rb_ary_slice used to add at the end of array, cause rb_ary_concat use rb_ary_slice.

Revision 37583 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: use shared array in rb_ary_slice

- array.c (rb_ary_splice): use shared array in rb_ary_slice. [Feature #6638]
  - use ary_ensure_room_for_push when rb_ary_slice used to add at the end of array, cause rb_ary_concat use rb_ary_slice.

Revision 37583 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: use shared array in rb_ary_slice

- array.c (rb_ary_splice): use shared array in rb_ary_slice. [Feature #6638]
  - use ary_ensure_room_for_push when rb_ary_slice used to add at the end of array, cause rb_ary_concat use rb_ary_slice.

Revision 37583 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: use shared array in rb_ary_slice

- array.c (rb_ary_splice): use shared array in rb_ary_slice. [Feature #6638]
  - use ary_ensure_room_for_push when rb_ary_slice used to add at the end of array, cause rb_ary_concat use rb_ary_slice.

Revision 37583 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: use shared array in rb_ary_slice

- array.c (rb_ary_splice): use shared array in rb_ary_slice. [Feature #6638]
  - use ary_ensure_room_for_push when rb_ary_slice used to add at the end of array, cause rb_ary_concat use rb_ary_slice.

Revision 37583 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: use shared array in rb_ary_slice

- array.c (rb_ary_splice): use shared array in rb_ary_slice. [Feature #6638]
  - use ary_ensure_room_for_push when rb_ary_slice used to add at the end of array, cause rb_ary_concat use rb_ary_slice.

Revision fdbd3716 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: speedup Array#unshift by using space in shared array

- array.c: speedup Array#unshift by using space in shared array. [Feature #6638]
  - when array owns its shared array (ARY_SHARED_NUM == 1), and there is enough space then try unshift values directly into shared array.

- when resulting array is big (~>64 items) then make it shared with enough room for future #unshifts, and then insert into shared array.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37584 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37584 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: speedup Array#unshift by using space in shared array

- array.c: speedup Array#unshift by using space in shared array. [Feature #6638]
  - when array owns its shared array (ARY_SHARED_NUM == 1), and there is enough space then try unshift values directly into shared array.
  - when resulting array is big (~>64 items) then make it shared with enough room for future #unshifts, and then insert into shared array.

Revision 37584 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: speedup Array#unshift by using space in shared array

- array.c: speedup Array#unshift by using space in shared array. [Feature #6638]
  - when array owns its shared array (ARY_SHARED_NUM == 1), and there is enough space then try unshift values directly into shared array.
  - when resulting array is big (~>64 items) then make it shared with enough room for future #unshifts, and then insert into shared array.

Revision 37584 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: speedup Array#unshift by using space in shared array

- array.c: speedup Array#unshift by using space in shared array. [Feature #6638]
  - when array owns its shared array (ARY_SHARED_NUM == 1), and there is enough space then try unshift values directly into shared array.
  - when resulting array is big (~>64 items) then make it shared with enough room for future #unshifts, and then insert into shared array.

Revision 37584 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: speedup Array#unshift by using space in shared array

- array.c: speedup Array#unshift by using space in shared array. [Feature #6638]
  - when array owns its shared array (ARY_SHARED_NUM == 1), and there is enough space then try unshift values directly into shared array.
  - when resulting array is big (~>64 items) then make it shared with enough room for future #unshifts, and then insert into shared array.

Revision 37584 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: speedup Array#unshift by using space in shared array

- array.c: speedup Array#unshift by using space in shared array. [Feature #6638]
  - when array owns its shared array (ARY_SHARED_NUM == 1), and there is enough space then try unshift values directly into shared array.
  - when resulting array is big (~>64 items) then make it shared with enough room for future #unshifts, and then insert into shared array.

Revision 37584 - 11/09/2012 07:08 AM - nobu (Nobuyoshi Nakada)

array.c: speedup Array#unshift by using space in shared array

- array.c: speedup Array#unshift by using space in shared array. [Feature #6638]
  - when array owns its shared array (ARY_SHARED_NUM == 1), and there is enough space then try unshift values directly into shared array.
  - when resulting array is big (~>64 items) then make it shared with enough room for future #unshifts, and then insert into shared array.

History

#1 - 06/25/2012 12:35 PM - matz (Yukihiro Matsumoto)

If it doesn't change the behavior, I am positive to merge this.

Matz.

#2 - 06/25/2012 03:35 PM - funny_falcon (Yura Sokolov)

I've fixed error introduced in second commit on the case x.concat(x).

#3 - 06/26/2012 06:00 PM - funny_falcon (Yura Sokolov)

This patch actually fixes non-linear Queue behaviour considered in http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/391324

#4 - 06/26/2012 07:38 PM - funny_falcon (Yura Sokolov)

I've add a couple commits to https://github.com/ruby/ruby/pull/133 and now Array becomes fully useful as a deque:

- #shift behaves old way: used shared array for circular buffer
- #push now knows if array is shared, and when ARY_SHARED_NUM(shared) == 1 then tries to push entries directly into shared array. So that, there is no additional allocations or memmove for most of pushes
- #pop behaves almost in old way, but it clears shared array element as #shift do.

- #unshift tries to unshift elements to shared array when ARY_SHARED_NUM(shared) == 1, and most of time provides a room for future unshifts.
- In case when array is not shared and it is large enough (currently, more than 116 elements), #unshift makes it shared and ensures that there is a room for future unshifts.

Full diff https://github.com/ruby/ruby/pull/133.diff
Patch (separate commits) https://github.com/ruby/ruby/pull/133.patch
Tests and patch against 1.9.3 https://gist.github.com/2981959

#5 - 06/27/2012 04:43 PM - funny_falcon (Yura Sokolov)

Another couple of commits to https://github.com/ruby/ruby/pull/133 :

- fixed error in rb_ary_cat_to_shared which leads to memory corruption
- avoid pathological performance cases (when array length is close to capacity) by increasing capacity earlier and leaving more room for future values

#6 - 07/14/2012 06:41 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to nobu (Nobuyoshi Nakada)

#7 - 09/02/2012 09:26 PM - funny_falcon (Yura Sokolov)

I've update branch a bit, cause previous code leads to some performance degradation for average application (which do not use array as queue).

New code does not hurts performance of average application and still provides great performance for array used as a queue.
Pull request: https://github.com/ruby/ruby/pull/174
Patch: https://github.com/ruby/ruby/pull/174.patch

#8 - 11/04/2012 11:44 AM - jonforums (Jon Forums)

On Win7 32bit with a one-line tweak to Yura's 174.patch to allow it to apply to trunk I get the following success when building with mingw-w64 gcc 4.7.2:

C:\Jenkins\workspace\ruby-trunk-svn>svn log -l 1

r37463 | nobu | 2012-11-03 21:19:11 -0400 (Sat, 03 Nov 2012) | 5 lines

C:\Jenkins\workspace\ruby-trunk-svn>svn st
M   array.c
?   array_as_queue_trunk.patch

sh-3.1$ make test-all
...
Finished tests in 620.647090s, 18.4727 tests/s, 4309.2106 assertions/s.
11465 tests, 2674499 assertions, 0 failures, 0 errors, 83 skips

ruby -v: ruby 2.0.0dev (2012-11-04 trunk 37463) [i386-mingw32]

sh-3.1$ make test
...
sample/test.rb:gc OK 4

test succeeded
PASS all 957 tests
./miniruby.exe -I../../../../Jenkins/workspace/ruby-trunk-svn/lib -I. -I.ext/common ../../../../Jenkins/workspace/ruby-trunk-svn/tool/runruby.rb --extout=.ext
-- --disable-gems "../../../../Jenkins/workspace/ruby-trunk-svn/bootstraptest/runner.rb" --ruby="ruby.exe"
../../../../Jenkins/workspace/ruby-trunk-svn/KNOWNBUGS.rb
2012-11-03 22:39:19 -0400
Driver is ruby 2.0.0dev (2012-11-04 trunk 37463) [i386-mingw32]
Target is ruby 2.0.0dev (2012-11-04 trunk 37463) [i386-mingw32]

KNOWNBUGS.rbPASS 0
No tests, no problem

#9 - 11/04/2012 07:51 PM - funny_falcon (Yura Sokolov)

I've rebased branch against trunk.

#10 - 11/09/2012 04:08 PM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r37581.
Yura, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

---

array.c: steal shared array's container when ARY_SHARED_NUM == 1

- array.c (rb_ary_modify): steal shared array's container when ARY_SHARED_NUM == 1. [Feature #6638]
  - Do not allocate new memory in rb_ary_modify when ARY_SHARED_NUM == 1 and length almost same.
  - Store ARY_CAPA instead of RARRAY_LEN in ary_make_shared, to make it useful.
  - Fix rb_ary_sort_bang accordantly.