Ruby - Feature #15010

Reduce allocation for rest parameters

08/19/2018 04:25 PM - chopraanmol1 (Anmol Chopra)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

Currently multiple arrays are allocated while making a call to method with rest parameter.

E.g.

```
def rest_method(*args) #-> This will create 2 arrays
end

def post_method(*args,last) #-> This will create 3 arrays
end
```

Applying following set of changes will reduce creation of array to 1

https://github.com/ruby/ruby/pull/1935

Benchmark Result:

trunk

```
                                user     system      total        real
benchmark_method            0.340000   0.000000   0.340000 (  0.337035)
rest_method                 0.964000   0.000000   0.964000 (  0.964660)
lead_method                 0.976000   0.000000   0.976000 (  0.976011)
post_method                 2.424000   0.000000   2.424000 (  2.421732)
lead_post_method            1.800000   0.000000   1.800000 (  1.799500)
rest_with_named_parameter   2.040000   0.000000   2.040000 (  2.040323)
lead_proc underflow_args    1.224000   0.000000   1.224000 (  1.225237)
opt_post_proc overflow_args 1.056000   0.000000   1.056000 (  1.057402)
```

modified

```
                                user     system      total        real
benchmark_method            0.336000   0.000000   0.336000 (  0.336911)
rest_method                 0.708000   0.000000   0.708000 (  0.706142)
lead_method                 0.720000   0.000000   0.720000 (  0.717971)
post_method                 1.896000   0.000000   1.896000 (  1.894426)
lead_post_method            1.560000   0.000000   1.560000 (  1.560495)
rest_with_named_parameter   1.464000   0.000000   1.464000 (  1.467313)
lead_proc underflow_args    0.864000   0.000000   0.864000 (  0.863980)
opt_post_proc overflow_args 0.772000   0.000000   0.772000 (  0.770364)
```

Associated revisions

Revision 1f4efb9aedfb8f537630f7c13e431bb230bebd31 - 08/28/2018 07:06 AM - ko1 (Koichi Sasada)

rest parameter optimization [Feature #15010]

- vm_args.c: rb_ary_dup(args->rest) to be used at most once during
  parameter setup. [Feature #15010]
  A patch by chopraanmol1 (Anmol Chopra) chopraanmol1@gmail.com.

- array.c (rb_ary_behead): added to remove first n elements.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64583 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 1f4efb9aedfb8f537630f7c13e431bb230bebd31 - 08/28/2018 07:06 AM - ko1 (Koichi Sasada)

rest parameter optimization [Feature #15010]

- vm_args.c: rb_ary_dup(args->rest) to be used at most once during
  parameter setup. [Feature #15010]
  A patch by chopraanmol1 (Anmol Chopra) chopraanmol1@gmail.com.

- array.c (rb_ary_behead): added to remove first n elements.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64583 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 1f4efb9a - 08/28/2018 07:06 AM - ko1 (Koichi Sasada)

rest parameter optimization [Feature #15010]

- vm_args.c: rb_ary_dup(args->rest) to be used at most once during
  parameter setup. [Feature #15010]
  A patch by chopraanmol1 (Anmol Chopra) chopraanmol1@gmail.com.

- array.c (rb_ary_behead): added to remove first n elements.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64583 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 08/20/2018 05:13 AM - mame (Yusuke Endoh)

Looks good to me. Though destructive operation to the rest array may make the source code unclear, performance is more important in this case, I think.

Some other functions in vm_args.c also use rb_ary_dup. There may be more room to optimize.

#2 - 08/20/2018 06:29 AM - chopraanmol1 (Anmol Chopra)

mame (Yusuke Endoh) wrote:

> Some other functions in vm_args.c also use rb_ary_dup. There may be more room to optimize.

Yes, it can be further optimized for keyword argument and argument setup for the block. I'll modify the patch in a day or two.

#3 - 08/20/2018 06:42 AM - normalperson (Eric Wong)

chopraanmol1@gmail.com wrote:

> Yes, it can be further optimized for keyword argument and argument setup for the block. I'll modify the patch in a day or two.

Cool! It's probably worth implementing something like
rb_ary_shift_m (but without the return value) to avoid looping
on rb_ary_shift.

#4 - 08/20/2018 10:22 AM - chopraanmol1 (Anmol Chopra)

- File bench_method_arg.rb added

- File improve_rest_parameters_setup_20_8_2018.patch added

- File deleted (bench_method_arg.rb)

- Description updated

#5 - 08/20/2018 10:27 AM - chopraanmol1 (Anmol Chopra)

normalperson (Eric Wong) wrote:

> Cool! It's probably worth implementing something like
> rb_ary_shift_m (but without the return value) to avoid looping
> on rb_ary_shift.

Added rb_ary_clear_m (suggestion for a better name will be appreciated) with suggested changes.

mame (Yusuke Endoh) wrote:

> Some other functions in vm_args.c also use rb_ary_dup. There may be more room to optimize.

Modified patch to ensure rb_ary_dup is called at most once.

#6 - 08/20/2018 11:22 AM - chopraanmol1 (Anmol Chopra)

- File 0001-Reduce-allocation-for-rest-parameters.patch added

- File deleted (improve_rest_parameters_setup.patch)

- File deleted (improve_rest_parameters_setup_20_8_2018.patch)

#7 - 08/20/2018 07:04 PM - normalperson (Eric Wong)

chopraanmol1@gmail.com wrote:

> normalperson (Eric Wong) wrote:
>
> > Cool! It's probably worth implementing something like
> > rb_ary_shift_m (but without the return value) to avoid looping
> > on rb_ary_shift.
>
> Added rb_ary_clear_m (suggestion for a better name will be
> appreciated) with suggested changes.

Thanks; for internal functions the name isn't as important :)

New functions prototypes should go into internal.h, though.
ruby/intern.h ended up being part of the public API and external
libraries depend on it :<

There's no reason for arg_rest_dup to be a macro instead of a
static inline function. Static inlines are preferred because
they make life easier for the compiler and debugger.

Also, multi-line macros without "do {} while (0)" is dangerous
to control flow.

Thanks again.

#8 - 08/20/2018 10:15 PM - mame (Yusuke Endoh)

Thank you, too. Two points:

First, the prefix _m is often used for an entry function of Ruby-level method that is passed to rb_define_method. Though it is just an internal function, it would be better to avoid the prefix. How about rb_ary_remove_first?

Second, I agree with ensuring rb_ary_dup is called at most once. But I'm afraid if rewriting the array without dup may cause obscure incompatibility. It is difficult for me to review your patch.

@ko1 (Koichi Sasada), could you review this?

#9 - 08/20/2018 11:14 PM - nobu (Nobuyoshi Nakada)

normalperson (Eric Wong) wrote:

> > Added rb_ary_clear_m (suggestion for a better name will be
> > appreciated) with suggested changes.
>
> Thanks; for internal functions the name isn't as important :)

What about rb_ary_clear_head? (or rb_ary_behead :)

#10 - 08/21/2018 04:41 AM - chopraanmol1 (Anmol Chopra)

- File 0001-Reduce-allocation-for-rest-parameters.patch added

- File deleted (0001-Reduce-allocation-for-rest-parameters.patch)

#11 - 08/21/2018 04:46 AM - chopraanmol1 (Anmol Chopra)

normalperson (Eric Wong) wrote:

> New functions prototypes should go into internal.h, though.
> ruby/intern.h ended up being part of the public API and external
> libraries depend on it :<
>
> There's no reason for arg_rest_dup to be a macro instead of a
> static inline function. Static inlines are preferred because
> they make life easier for the compiler and debugger.

Updated.

mame (Yusuke Endoh) wrote:

> First, the prefix _m is often used for an entry function of Ruby-level method that is passed to rb_define_method. Though it is just an internal function, it would be better to avoid the prefix. How about rb_ary_remove_first?

For now, I'm renaming the method to rb_ary_behead (suggested by nobu)

#12 - 08/21/2018 05:19 AM - chopraanmol1 (Anmol Chopra)

I'm also thinking of an alternate solution which will avoid passing the skip_dup_flag variable around, If we can ensure that args->rest is not used/assigned until args_copy is called. To do this when VM_CALL_ARGS_SPLAT flag is on instead of assigning args->rest we could expand the splat arg to locals / args->argv.

Unless it breaks test beyond repair, I'll add this alternate patch with the respective benchmark(It probably will be slower for the large array), so it can be compared side by side. In this solution, args_setup_post_parameters can be further modified to use args->argv instead of args->rest which makes zero allocation for the following example:

```
def opt_post(a,b,c=1,d=2,e,f); end
```

#13 - 08/21/2018 05:36 AM - chopraanmol1 (Anmol Chopra)

normalperson (Eric Wong) wrote:

> New functions prototypes should go into internal.h, though.
> ruby/intern.h ended up being part of the public API and external
> libraries depend on it :<

Moving method to internal.h breaks jit https://travis-ci.org/ruby/ruby/builds/418529271 , I'm not sure how to fix this failure.

#14 - 08/21/2018 05:59 AM - nobu (Nobuyoshi Nakada)

chopraanmol1 (Anmol Chopra) wrote:

> Moving method to internal.h breaks jit https://travis-ci.org/ruby/ruby/builds/418529271 , I'm not sure how to fix this failure.

Define the function with MJIT_FUNC_EXPORTED.

#15 - 08/21/2018 07:26 AM - chopraanmol1 (Anmol Chopra)

chopraanmol1 (Anmol Chopra) wrote:

> I'm also thinking of an alternate solution which will avoid passing the skip_dup_flag variable around, If we can ensure that args->rest is not used/assigned until args_copy is called. To do this when VM_CALL_ARGS_SPLAT flag is on instead of assigning args->rest we could expand the splat arg to locals / args->argv.

Implementation: https://github.com/ruby/ruby/compare/trunk...chopraanmol1:improve_rest_parameters_setup_v2

Benchmark result

```
                              trunk        patch 1       patch 2
benchmark_method              0.196346     0.197841      0.196466
rest_method                   0.788287     0.539768      0.535512
lead_method                   0.792892     0.547752      0.533818
post_method                   1.133035     0.636972      0.540609
lead_post_method              0.867869     0.709440      0.370182
benchmark_method *args        0.227389     0.230066      0.227671
rest_method *args             0.826490     0.559881      0.563779
lead_method *args             0.821036     0.602590      0.565583
```

```
post_method *args               1.157621     0.649459      0.570189
lead_post_method *args          1.064632     0.687248      0.387054
rest_method *long_args          0.985696     0.766369      0.779729
lead_method *long_args          0.997824     0.870107      0.794615
post_method *long_args          1.703731     0.863923      0.813282
lead_post_method *long_args     1.707543     0.989116      0.802757
rest_with_named_parameter       1.862414     1.293406      1.255951
bench proc                      0.275176     0.263893      0.260555
lead_proc underflow_args        1.149043     0.801893      0.363017
opt_post_proc overflow_args     1.025754     0.717966      0.312920
```

chopraanmol1 (Anmol Chopra) wrote:

> In this solution, args_setup_post_parameters can be further modified to use args->argv instead of args->rest which makes zero allocation for the following example:

args->argv and locals are pointing to same address so it is not feasible.

Note: This second patch is not the final implementation, there are few more changes. In second patch args_check_block_arg0/args_setup_opt_parameters function can still assign args->rest / modify args->rest_index, I'll look into this later (only if the second patch seems more acceptable over first) if it can be completely avoided. In case it can be avoided most of the method handling args->rest can be cleaned after that, which will also ensure that args->rest_index is never modified. As a result, we could even avoid calling rb_ary_behead.

#16 - 08/21/2018 07:35 AM - chopraanmol1 (Anmol Chopra)

- File Reduce-allocation-for-rest-parameters-v2.patch added

- File Reduce-allocation-for-rest-parameters-v1.patch added

- File bench_method_arg_v2.rb added

- File deleted (0001-Reduce-allocation-for-rest-parameters.patch)

#17 - 08/21/2018 12:45 PM - chopraanmol1 (Anmol Chopra)

Limitation of patch 2.

1. Patch 2 gets slower than Patch 1 for a large array. Array with length 100 - 200 have similar performance but beyond that patch 1 is faster in most of the case.
2. Patch 2 results in segmentation fault for following:
   https://github.com/ruby/ruby/blob/8e66ffc1d756c42ee025a56672ad71f2200ca6be/test/ruby/test_method.rb#L951

~~Ignoring above limitation patch 2 do perform better for the small array. One Hack Solution can be to check the length of splat arg against some arbitrary number to decide if splat arg should be expanded to args->argv or should be assigned to args->rest. But it doesn't sound like a nice solution.~~

I'm not able to reproduce benchmark result for Patch 2. Even for splat args with size under 100 patch 2 has similar performance to patch 1. Patch 2 is only significantly faster on **lead_proc underflow_args** benchmark. Given the above limitation patch 2 is not worth it.

#18 - 08/23/2018 07:22 AM - ko1 (Koichi Sasada)

sorry, which patch should I review?

#19 - 08/23/2018 07:41 AM - chopraanmol1 (Anmol Chopra)

ko1 (Koichi Sasada) wrote:

> sorry, which patch should I review?

Reduce-allocation-for-rest-parameters-v1.patch

#20 - 08/27/2018 05:22 AM - chopraanmol1 (Anmol Chopra)

- File Reduce-allocation-for-rest-parameters-v1.patch added

#21 - 08/27/2018 05:22 AM - chopraanmol1 (Anmol Chopra)

- File deleted (Reduce-allocation-for-rest-parameters-v1.patch)

#22 - 08/27/2018 05:26 AM - chopraanmol1 (Anmol Chopra)

@ko1 (Koichi Sasada), It would be great if you could review
https://bugs.ruby-lang.org/attachments/7343/Reduce-allocation-for-rest-parameters-v1.patch

#23 - 08/27/2018 06:33 AM - ko1 (Koichi Sasada)

Sorry for late response.

Idea (as my understanding)

~a rest parameter" is dup multiple times because of current implementation. Only 1 "dup" is needed. They should be eliminate.
The patch try to manage "dup'ed or not" by passing skip_rest_ary_dup, and if it is true, then we don't need to dup the rest parameter again.

Comment

I'm fine to introduce your idea.
Why don't you put a new field in args_info?

#24 - 08/27/2018 07:41 AM - chopraanmol1 (Anmol Chopra)

ko1 (Koichi Sasada) wrote:

> # Idea (as my understanding)
>
> ~a rest parameter" is dup multiple times because of current implementation. Only 1 "dup" is needed. They should be eliminate.
> The patch try to manage "dup'ed or not" by passing skip_rest_ary_dup, and if it is true, then we don't need to dup the rest parameter again.

Yes, and once a rest parameter is duped it mutates the array in case if rest_index is modified (Previously, only args_setup_post_parameters used to mutate rest parameter).

> # Comment
>
> I'm fine to introduce your idea.
> Why don't you put a new field in args_info?

This suggestion makes a lot of sense as it will simplify this patch, I'll update the patch soon to reflect this.

#25 - 08/27/2018 09:37 AM - chopraanmol1 (Anmol Chopra)

- File Reduce-allocation-for-rest-parameters-v1.patch added

- File deleted (Reduce-allocation-for-rest-parameters-v1.patch)

#26 - 08/27/2018 10:08 AM - chopraanmol1 (Anmol Chopra)

@ko1 (Koichi Sasada), I've added new field rest_dupped to args_info.

Updated patch https://bugs.ruby-lang.org/attachments/7344/Reduce-allocation-for-rest-parameters-v1.patch

#27 - 08/28/2018 06:53 AM - ko1 (Koichi Sasada)

It seems fine.
I'll commit it.

#28 - 08/28/2018 07:06 AM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

Applied in changeset trunk|r64583.

---

rest parameter optimization [Feature #15010]

- vm_args.c: rb_ary_dup(args->rest) to be used at most once during
  parameter setup. [Feature #15010]
  A patch by chopraanmol1 (Anmol Chopra) chopraanmol1@gmail.com.
- array.c (rb_ary_behead): added to remove first n elements.

Files

bench_method_arg.rb	1.32 KB	08/20/2018	chopraanmol1 (Anmol Chopra)

Reduce-allocation-for-rest-parameters-v2.patch	3.57 KB	08/21/2018	chopraanmol1 (Anmol Chopra)
bench_method_arg_v2.rb	3.15 KB	08/21/2018	chopraanmol1 (Anmol Chopra)
Reduce-allocation-for-rest-parameters-v1.patch	5.43 KB	08/27/2018	chopraanmol1 (Anmol Chopra)