Ruby trunk - Feature #15602

Eliminate recording full-width hash value for small Hash

02/13/2019 10:05 AM - ko1 (Koichi Sasada)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Abstract

Let's shape up small hash value (1 to 8 entries) from 192B to 128B on 64bit ptr environments.

Data structure proposal

(step 1) Record only key and value pairs.

Now Ruby 2.6, 1 to 8 entry Hash objects allocate 192 byte (8B * 3 (key, value and hash value triple) * 8 entry = 192B) with ar_table (instead of st_table).
Eliminating to record hash value will reduce this allocation from 192B to 128B (8 * 2 * 8).

(step 2) 1 byte hash value

For 1 to 8 entries, full-width Hash value (8 bytes) may be too long to lookup the entry.
1 byte hash value can be generated from 8 byte hash value.
(hash_value & 0xff is most simple way to get it, but not sure it is enough)

Name 1 byte hash value as "hash hint" on my patch.

(step 3) Embed hash hint into RHash

RHash::iter_lev is used to recognize nesting level of a hash (h.each{ "h's iter_lev is 1 here" }).
However, there are only a few cases that this value is bigger than 1.
So we can put this value into flags (if it exceeds the limit, we can store this value into hidden attribute).
8 hash hints becomese 8B == sizeof(VALUE), so we can embed this value into RHash.

Discussion

- Pros.
  - We can reduce allocation size of small Hash.
  - Increase cache locality on hash lookup
    - We don't need to touch ar_table (allocate memory) if hash hints doesn't match.
    - We can access correct ar_table entry directly.
- Cons.
  - hash hints can conflict more than full-width hash value => may increase eql? call.
    - performance down
    - incompatibility

Evaluation

I tested this patch and it slightly increase performance (not so big, on my micro-benchmark).
Memory consumption is reduced theoretically.

Patch

https://github.com/ko1/ruby/tree/hash_small_ar

History

#1 - 02/13/2019 08:27 PM - Eregon (Benoit Daloze)

IIRC, not storing #hash breaks specs, does it pass test-spec?
Maybe the 8-bit #hash is enough to avoid problems?

#2 - 02/14/2019 08:18 AM - ko1 (Koichi Sasada)

- Description updated

Eregon (Benoit Daloze) wrote:

> IIRC, not storing #hash breaks specs, does it pass test-spec?

Fortunately, no problem.

> Maybe the 8-bit #hash is enough to avoid problems?

I'm not sure what "the 8-bit #hash" is.

#3 - 02/15/2019 11:51 AM - Eregon (Benoit Daloze)

ko1 (Koichi Sasada) wrote:

> I'm not sure what "the 8-bit #hash" is.

The same as "1 byte hash value".
i.e. after step 1 I would expect tests/specs to fail, but probably the "1 byte hash value" is enough to fix them.

#4 - 03/21/2019 02:42 AM - mame (Yusuke Endoh)

Eregon (Benoit Daloze) wrote:

> The same as "1 byte hash value".
> i.e. after step 1 I would expect tests/specs to fail, but probably the "1 byte hash value" is enough to fix them.

I think so. I'm unsure how may people encounter this incompatibility.

```
class Foo
  def hash
    $hash
  end
end

obj = Foo.new
h = {}
$hash = 0
h[obj] = 42
$hash = 256
p h[obj] #=> nil in trunk, 42 in patched
```